CHAPTER 1

GENERAL DESCRIPTION AND OPERATION

1.0 General Description

The MEK6802D3 board (see Figure 1.0.1) is designed to be a single board microcomputer, for use in both the educational environment and for computer based system applications.

This system employes the MC6802 microcomputer chip. It also contains 256 bytes of user RAM (MCM6810) and 128 bytes of operating system (OS) stack. The operating system software (called D3BUG) is contained in the 2K bytes of Read Only Memory (ROM) storage of the MC6846. The keyboard and eight seven-segment displays are used in conjunction with D3BUG to enter and debug user programs. The I/O port of the MC6846 is accessible through a 16 pin socket for external control of user defined applications such as timers, relay control, etc.

Provisions are made for system expansion through the system bus. The D3BUG operating system is expanded by 2K bytes (MCM68316) to accommodate additional features and commands necessary for the expanded system. Also the system bus has additional signals to support eight levels of memory paging. Hence, the expanded operating monitor with 100% memory decoding and paging can make the MEK6802D3 a very flexible and powerful computer system.

1.1 Address Space (stand-alone computer)

The address space organization of the MEK6802D3 computer board is given in the address map (see Figure 1.1.1). Chapter 3 gives a functional

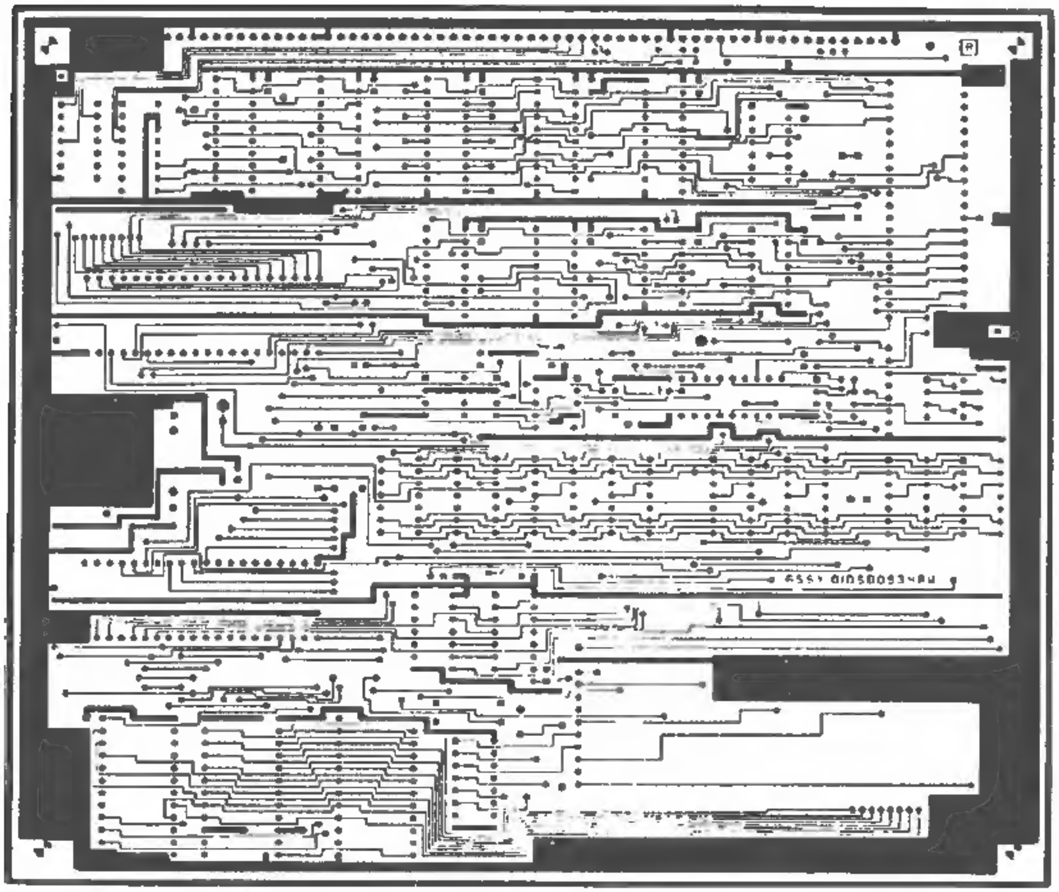


FIGURE 1.0.1. MEK6802D3 BOARD

$FFFF – $F800	MC6846 ROM	D3BUG
$81FF – $8180	MCM6810 RAM	USER STACK OPTIONAL USER STACK - RELOCATABLE FROM $0080
$817F – $8100	MCM6810 RAM	D3BUG OPERATING SYSTEM STACK
$808B – $8088	MC6821	KEYPAD/DISPLAY PIA
$8087 – $8080	MC6846	I/O - TIMER
$00FF – $0080	MCM6810 RAM	USER STACK - RELOCATABLE TO $8180
$007F – $0000	MC6802 RAM	USER STACK - CAN BE DISABLED WHEH USING OFF-BOARD MEMORY

FIGURE 1.1.1. ADDRESS MAP

1.1 Address Space (stand-alone computer)(cont'd)

description for the address space configuration for an expanded system.

1.2 Firmware Features

The 2K firmware monitor for the MEK6802D3 microcomputer system, D3BUG, resides in the MC6846. The function of the monitor is to provide a means for communication with and control of the system microprocessor by using the keyboard and display. D3BUG has the following capabilities and features:

1. Examine and change memory locations with ability to increment to next location or decrement to previous location.
2. Automatically verify memory change by displaying both the digits entered and the actual contents of memory.
3. Display and change MPU registers.
4. Calculate offsets for branch instructions.
5. Single step trace of programs in both RAM and ROM.
6. Set, clear, and examine up to 8 breakpoints.
7. * Punch designated memory to audio cassette using the expansion I/O board, MEK68IO, at either 300 or 1200 baud rate.
8. * Load cassette tape into memory.
9. * Verify tape after punching or loading.
10. Go to and execute user program.
11. Abort user program.
12. Execute any keyboard control functions during execution of user program.
13. User definition of each of the hexadecimal keys to correspond to a user program.

1.2 Firmware Features (cont'd)

14. Optional user definition of interrupt vectors.

15. Access to monitor subroutines for user programs.

* Features 7, 8, and 9 can only be used in an expanded system with a MEK6810 board.

1.3 Preparation For Use

The MEK6802D3 requires a single +5 V power supply (5% tolerance) with a maximum current capacity of 0.9 amps. The kit comes with a 5 pin contact wafer assembly which can be used to attach the power through the PC board connector. Two wires (at least 22 gauge) must be soldered to the contact wafer assembly as shown in Figure 1.3.1.

The 5 pin contact wafer assembly pins are accessable at both ends and the two wires are soldered to the short pin side. The longer pin side is mated to the PC board connector (pins 1-5) as noted in Figure 1.3.1. To insure that the soldered wires do not short to other pins, it is recommended that sleeving is used on the two soldered pins.

Before the power supply is connected to the PC board, be sure that the surface that is to support the D3 card is a non-conductive material such as wood. Next, make sure that the power supply is off before connecting the connector assembly to the PC board.

1.4 Start Up Procedure

When the power supply is switched on, a prompt sign (=) should appear in the left most display (marked 1 in Figure 1.3.1) and the remaining displays should be blank. If the display does not come up in this mode, press

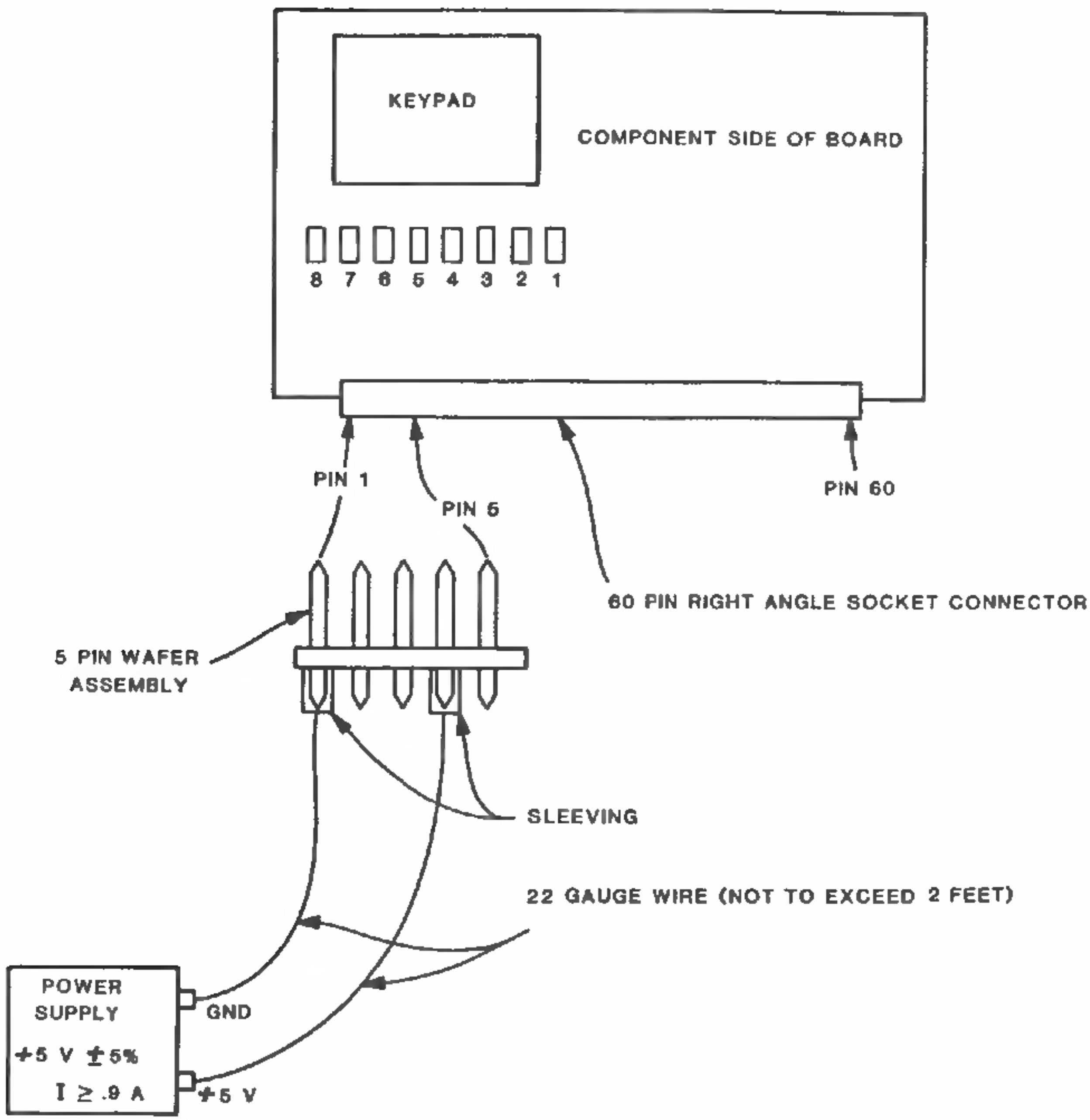


FIGURE 1.3.1. POWER CONNECTION

1.4 Start Up Procedure (cont'd)

the RS button on the keypad. When RS is depressed momentarily, the display should go to the state described when power was first turned on. If the display registers something different, then turn off the power supply and check for proper connection and continuity of the power connector. Also check to make certain that the power supply is within the specified +5 V ± 5% range.

When the power is on and all displays blank except the prompt sign (=), the monitor (D3BUG) firmware is now in operation and any of the functions described in the next section may be invoked by means of the data and command keys of the keypad.

1.5 Operating Procedure

The keypad/display, in conjunction with D3BUG provides examine operations for the computer and entering and trouble-shooting of programs. The keypad has sixteen keys labeled Ø - F for entry of hexadecimal data and nine keys for commanding the following functions.

RS	-	System Reset
M	-	Memory Display
GO	-	Go
EX	-	Escape
FS	-	Special Function
T/B	-	Trace/Breakpoint
RD	-	Register Display
P/L	-	Punch/Load
FC	-	Clear Special Function

1.5 Operating Procedure (cont'd)

Operating procedures for each of these functions are described in the following paragraphs with the exception of the P/L operation which will be covered in detail in Chapter 3.

RS : Reset key re-initializes the monitor and places a (=) prompt on the display.

M : The memory display key is used to display and alter data stored in memory. The function is invoked by entering the hex address and depressing the M key. The first four displays provide the address. The middle two displays will contain any hex numbers which may be entered by the user. The far right two displays will show what is stored at that address. If the address is not in RAM, the far right display may not be the same as the middle display. To step towards high memory, type GO and the next higher memory location will be displayed. To examine a previous memory location, type M and the previous memory location will be displayed.

The memory display function contains a subfunction which calculates offsets for 6800 branch instructions. To use it in the memory display function, type FS and the display 'A' appears prompting for an address. Enter the desired branch address and type GO. The display will return to memory display. If a legal offset was calculated, it will be displayed in both data displays and stored at the proper location. If an offset out of range was calculated, an FF will be displayed in the middle

two LEDs and the data in RAM (as displayed in the last two LEDs) will not change. If no RAM is at the displayed address, the offset will be displayed in the middle two LEDs only. To re-enter the monitor from the memory display, type EX and the prompt (=) will appear.

GO : The GO key allows the user to execute his programs. To use, enter the programs starting address and type GO. If any break-points have been set, they are placed into the user's program and are removed when a breakpoint is encountered. If the user wishes to abort type EX. If the program is aborted or ends in any other than executing a breakpoint, the breakpoints will not be removed.

EX : The escape key is used to re-enter the monitor from any of the commands or from the user's program. The EX key does not change the breakpoints that have been previously entered.

FS : Special Function Key converts all hex keys, P/L, and T/B to alter-nate function keys. The P/L key causes the load from tape operation to occur. The T/B key allows breakpoints to be entered. The hex keys are defined by the user who must create a jump address table and place the starting address of that table in UHASH ($81Ø2); a display '= FS' appears when FS is typed in the prompt mode. The $ symbol signifies that the number is hexadecimal.

1.5 Operating Procedure (cont'd)

FC : Clears the special function mode and removes the 'FS' from the display.

T/B: The Trace/Breakpoint key is used to cause single step traces or for Inserting/Displaying/Deleting breakpoints. The Trace function executes one instruction in a program and displays the registers at the end of execution. The registers may then be modified as desired. The user may trace through all the 6800's instructions including the SWI instruction. When an SWI is encountered, the the Trace continues in the user's software interrupt routines. This enables the user to test his interrupt program by placing a SWI instruction in his program and storing the interrupt address at USWIV ($81Ø8). Trace also allows the user to trace through a program stored in ROM. To use the Trace function, type RD, enter the address of the instruction in the program counter, and type T/B. The user may trace through as many instructions as are needed. If it is desired to execute the program from the display address, type GO. To return to the monitor type EX.

The breakpoint function allows the user to set up eight breakpoints and it allows the user to edit any breakpoints which already exist. Breakpoint addresses are stored in a sequential table along with the op-code at that address. Breakpoints are entered into the user programs when the GO key is typed and removed when a breakpoint point occurs. If any other method

is used to return to the monitor, i.e. a jump, the breakpoints will not be removed. To use the breakpoint function type FS and T/B. If this is the first time into the function, the display will be 'ØØØØ Ø' for no breakpoints. Enter the address of the desired breakpoint and type FS. For example: we wish to place a breakpoint at $5ØØØ. Type FS and T/B. The display is 'ØØØØ Ø', enter $5ØØØ and depress FS; the display is now '5ØØØ 1'. This may be repeated until an '8' is display in the far right side.

To display the breakpoints that have been set, type GO. The breakpoints will be displayed one at a time starting with the first breakpoint entered and will repeat until all the breakpoints have been displayed. When the breakpoint function is re-entered at a later time, the breakpoint that was entered first will be displayed first and the rest will be displayed in the order that they were entered.

To remove a breakpoint, step to the breakpoint that is to be removed with the GO key and type FC. The breakpoint entry is removed from the table, the table is compressed, and the breakpoint count decremented by one. One or all of the breakpoints may be cleared in this way. To clear all the breakpoints, push the RS (reset) key. Breakpoints may be set at any address in RAM memory. Breakpoints can not be used in a program stored in ROM.

RD: The register display key allows the user to display and modify the user's registers. To display the program counter, type RD and a display appears. This display breaks down as follows: the first four hex characters are the address in the program counter, the next two are the byte at that address, and PC stands for the program counter. The rest of the registers may be displayed one at a time by pressing the RD key. The registers are displayed in the following order:

Program Counter	(PC)
Index Register	(1d)
Accumulator A	(AA)
Accumulator B	(Ab)
Condition Code Register	(CC)
Stack Pointer	(SP)

The user may trace the instruction indicated by the PC by typing T/B. The next instruction will appear and the user remains in register display. The user may continue execution of his program by typing GO. To return to the monitor, type EX.

In order to better understand the keyboard commands of D3BUG, the following examples are given:

Example 1: Assume you want to examine the contents of address location $F957.

1.5 Operating Procedure (cont'd)

(1) Enter D3BUG by typing RS. The prompt should appear in the left most display with the remaining displays blank as illustrated.

Address Field				Key Data Field		Read Data Field	
=							

(2) Type F

0	0	0	F				

(3) Type 9

0	0	F	9				

(4) Type 5

0	F	9	5				

(5) Type 7

F	9	5	7				

(6) The desired address is now entered, and to display the contents of this address type M.

F	9	5	7	0	3	0	3

1.5 Operating Procedure (cont'd)

The contents of address F957 will always be Ø3 because this portion of the address space (see Figure 1.1.1) is the monitor program and it is in ROM. ROM contains permanent data and cannot be altered. To verify this, try to change the data at this location by entering data through the keypad. For example type key 5. The resulting display should be:

F957	Ø5	Ø3
Address Field	Key Data Field	Read Data Field

Note that the key data field is a Ø5. This data should have been written into the address specified by the address field. However, the read data field remained a Ø3 which meant that this location did not take the keypad data. Had this location taken the data, then the read data field would display a Ø5 instead of a Ø3.

Example 2: Using the previous example the GO key function can be examined. With the address and data field given:

F957	03	03

(1) Type GO

F958	7C	7C

The address was incremented by 1 and the data of this address is displayed.

(2) If GO is typed four more times, the resultant address and data fields should be:

1.5 Operating Procedure (cont'd)

F959	81	81
F95A	13	13
F95B	FE	FE
F95C	81	81

To examine the contiguous addresses below a selected address, all that is required is to type the M key.

Example 3: Using the address and data of example 1, the display will yield the following results when the M key is typed.

F956	24	24

Hence each time the M key is typed the address will decrement by 1 and the data of the new address displayed. Typing the M key four more times will yield the following results:

F955	14	14
F954	81	81
F953	67	67
F952	14	14

Example 4: This example will show how to examine and enter data in the user RAM; for this example lets examine the contents of location $ØØØ8.

1.5 Operating Procedure (cont'd)

(1) Return to monitor by typing EX.

=							

(2) Type in address 8.

Ø	Ø	Ø	8				

(3) To examine this address, type M.

Ø	Ø	Ø	8	?	?	?	?

The two data fields have question marks. The question marks are there, because unless previous data was stored into this address location, the data contents will be unknown. This unknown data state is a result of the storage elements in the RAM, (see Figure 1.1.1) having random data storage on power up.

(4) To store data (for this example use $8F), first type 8.

Ø	Ø	Ø	8	Ø	8	Ø	8

(5) Next type F.

Ø	Ø	Ø	8	8	F	8	F

Memory location $ØØØ8 will now contain the data $8F. This data will remain there, until it is purposely changed or power is turned off.

1.5 Operating Procedure (cont'd)

Example 5: The following example program is suitable for gaining familiarity with the D3BUG monitor features. The program adds the five values in locations $10 through $14 using Accum. A and stores the final results in location $15. The intermediate total is kept in Accum. A. Accum. B is used as a counter to count down the loop. The Index Register contains a "pointer" (i.e., X contains the address) of the next location to be added. The program, as follows, contains an error which will be used later to illustrate some of D3BUG's features.

In the following listing, the leftmost column contains the memory address where a byte (8 bits) of the program will be stored. The next column contains the machine language op-code and data for a particular microprocessor instruction. The next columns contain the mnemonic representation of the program in assembler format.

```
*
* Add 5 numbers at locations $10-$14.
* Put answer in location $15.
*

ØØ2Ø    8E    STRT    LDS   #$FF      DEFINE STACK IN USER AREA
ØØ21    00
ØØ22    FF
ØØ23    4F            CLRA            TOTAL =0
```

1.5 Operating Procedure (cont'd)

```
ØØ24    C6            LDAB  #4       INITIALIZE COUNTER
ØØ25    04
ØØ26    CE            LDX   #10      POINT X TO LOCATION 10
ØØ27    00
ØØ28    10
ØØ29    Ab    LOOP    ADDA  0,X      ADD 1 LOCATION TO TOTAL
ØØ2A    00
ØØ2b    08            INX            POINT X TO NEXT LOCATION
ØØ2C    5A            DECB           DONE ALL 5 LOCATIONS?
ØØ2d    26            BNE   LOOP     BRANCH IF NOT EQUAL
ØØ2E    FA
ØØ2F    97            STAA  $15      SAVE ANSWER
ØØ3Ø    15
ØØ31    7E            JMP PROMPT     GO TO D3BUG PROMPT
ØØ32    F8
ØØ33    57
```

A detailed procedure for entering and debugging this program is shown in the following steps.

1. Start Up and Enter the Program in RAM

   A. Turn power on. Type the RS button. D3BUG will respond with a (=).

   B. Type $ØØ2Ø followed by the M key. This displays the current contents of location $ØØ2Ø.

C. Type 8E. This replaces the contents of $ØØ2Ø with 8E which is the op-code for the first instruction.

D. Type GO. This steps to the next location ($ØØ21) and displays the contents.

E. Type ØØ.

F. Type GO.

G. Type next byte of op-code or operand (FF in this case).

H. Repeat steps F and G for remaining instructions.

I. Type EX. Abort input function.

2. Verify that the program was entered correctly.

   A. Type ØØ2ØM. Location $2Ø will be displayed.

   B. Type GO. Next location will be displayed.

   C. Repeat step B until done, visually verifying data entered in Step 1.

   D. Type EX.

3. Enter data in locations $10 - $14.

   A. Same as 1, except type ØØ1ØM to start the sequence. Any data may be entered; however, for purposes of this example Ø1, Ø2, Ø3, Ø4, and Ø5 should be entered.

   B. Type EX.

4. Verify Data.

   A. Repeat step 2 except type ØØ1ØM to begin the sequence. Verify

that the memory contains the values Ø1, Ø2, Ø3, Ø4, Ø5 in sequential order.

5. Run the Program.

   A. Type EX to insure no other option is active.

   B. Type ØØ2ØGO. The program will run down to the "JUMP" instruction at location $31 which will cause it to go to D3BUG and show (=) on the display.

6. Check the Answer.

   A. Type ØØ15M. (The answer is stored in location $15). Note that it says $ØA (decimal 1Ø). The correct answer is $ØF or decimal 15; therefore, there is a problem in the program as originally defined. The next steps should help isolate the problem and correct it.

7. Breakpoint and Register Display

   A. It might be helpful to see what the program was doing each time it went through the loop. Therefore, set a breakpoint at the beginning of the loop, location $ØØ29. To do this type EX, FS, and T/B. Next type the address ØØ29 and then FS.

   B. A breakpoint could also be set at location $ØØ2F. To do this type 2F and then FS.

   C. D3BUG must be told where to begin, so type EX and then ØØ2ØGO. The program will run to the breakpoint and then display ØØ29 Ab. At this point the program is suspended just before location 29

and is in D3BUG. On detecting this breakpoint, D3BUG automatically displays the PC and is in the register display mode.

D. Type RD (go to next register). The display should read ØØ1Ø. This is the value of the X Register.

E. Type RD-Display = ØØ (A Register).

F. Type RD-Display = Ø4 (B Register).

G. Type RD-Display = FØ (Condition Code Register).

H. Type RD-Display = ØØFF (Stack pointer).

I. Type RD-Display = ØØ29 PC. The register display is circular and steps D through H could be repeated.

J. Type GO (Proceed). Display will show ØØ29 Ab PC. Once again the registers may be examined.

K. Type GO (Proceed). Same comment as J.

L. Type GO (Proceed). Same comment as J.

M. Type GO (Proceed). Display will now type ØØ2F 97 PC. The program has now successfully completed the loop four times and the A accumulator contains the incorrect sum.

8. Correcting the Program.

A. From above it is evident that although the program was supposed to add five numbers, the loop was executed only four times. There fore, the LDAB #4 instruction at location $24 and $25 should have initialized B to five. There are two approaches to fix the problem; one is temporary, the other is permanent. First the temporary one:

1.5 Operating Procedure (cont'd)

B. Type EX.

C. Type FS, T/B, then type FC twice and this will clear existing breakpoints.

D. Type ØØ26FS. Set a breakpoint just after B register is loaded.

E. Type EX.

F. Type ØØ2ØGO. The program will execute up until $ØØ26 and then go to the breakpoint routine and, display 'ØØ26 CE PC'.

G. Type RD three times. This displays the current value in the B accumulator (Ø4 Ab).

H. Type Ø5. The display will change to Ø5 Ab.

I. Type GO. Program should execute and return to prompt (=).

J. Type ØØ15M. Display = ØØ15 ØF. The program has now calculated the correct value for the addition of the five numbers. This verifies the fix but would be inconvenient to do each time the program was executed. A permanent change would be:

K. Type EX, FS, and T/B. Then type FC once and this clears existing breakpoints.

L. Type EX.

M. Type ØØ25M. The display = ØØ25 Ø4.

N. Type Ø5. The display = ØØ25 Ø5. This will now permanently change the LDAB #4 instruction to a LDAB #5 instruction.

O. Type EX.

P. Type ØØ2ØGO. Execute the program.

Q. Type EX.

1.5 Operating Procedure (cont'd)

R. Type ØØ15M. Display = ØØ15 ØF, the expected answer; the program is permanently fixed.

9. Trace Through the Program.

A. Type EX. To trace from the beginning:

B. Type FS and T/B. Then type FC as many times as required to remove breakpoints.

C. Type FS and T/B. Next type ØØ2Ø and then FS. This sets a breakpoint at the first instruction.

D. Type EX.

E. Type ØØ2ØGO (Go to user program). D3BUG will immediately get the breakpoint and type ØØ2Ø 8E.

F. Type T/B. The program will execute one instruction and display ØØ23 4F. At this point the user can either display the registers by typing RD or continues. To continue:

G. Type T/B. GO to next instruction. Display register if desired.

H. Continue step G for as long as desired.

I. Type EX. Clear trace mode.

CHAPTER 2

HARDWARE DESCRIPTION

2.0 Central Processing Unit (CPU)

The MC6802 microcomputer chip is the CPU of the MEK6802D3 computer board. The MC6802 is a 6800 microprocessor with additional features; a clock oscillator circuit and 128 bytes of RAM. This powerful microcomputer chip contains an arithmetic logic unit (ALU), various registers, and control logic. It has a set of 72 different instructions, that include binary and decimal arithmetic logic, shift, rotate, fetch, store, branch interrupt, and stack manipulation. For more detailed information, see the MC6802 data sheet in the appendix. With coded instructions from memory, the MC6802 controls and manipulates address, data, and control information within the system. Thus, the MC6802 is the heart of the MEK6802D3 computer system and interfaces with the rest of the components on the board as illustrated in the system block diagram, Figure 2.0.1.

SYSTEM BUS: There are two buses on the computer card; the internal bus that interconnects components on the card and the external or system bus that connects the MEK6802D3 card with the system expansion cards.

The internal bus contains the following signals:

(1) 16 address lines
A∅ - A15

(2) 8 data lines
D∅ - D7

(3) 9 control lines

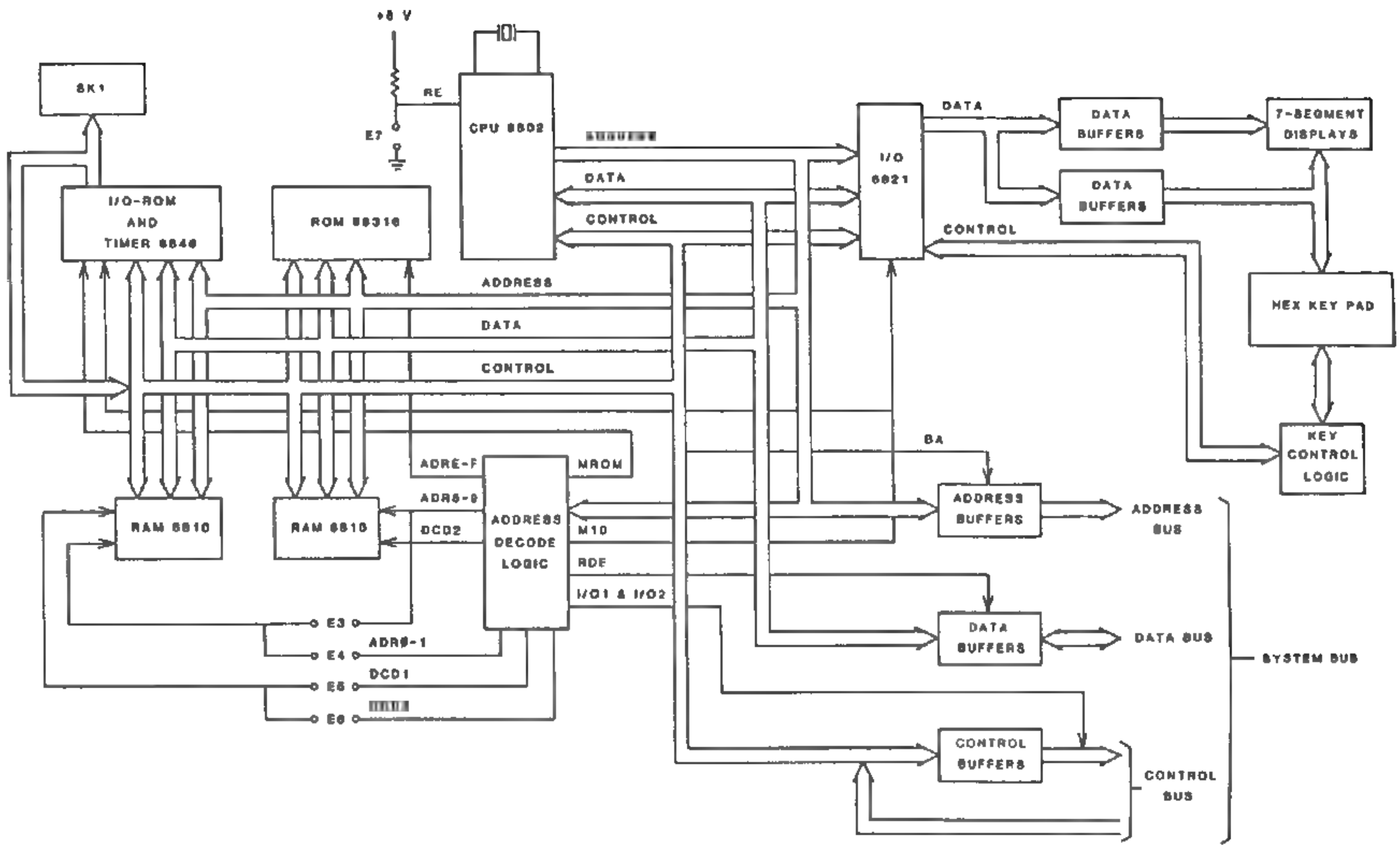


FIGURE 2.0.1. SYSTEM BLOCK DIAGRAM

(a) E - ENABLE - This signal is the system clock. A standard 3.579545 MHz crystal is used by the processor to generate the 894.8 KHz system clock.

(b) R/$\overline{\text{W}}$ - Read/Write - This is the read-write control line. Its logic state determines the direction of data (into or out of) a selected chip.

(c) VMA - Valid Memory Address - This control signal indicates that a valid address is on the address bus.

(d) MR - Memory Ready - This control signal can cause the E signal to be stretched. When MR is high, E will be in normal operation. When MR is low, E may be stretched an integral multiple of half periods, thus allowing interface to slow devices.

(e) $\overline{\text{RESET}}$ - this line is used to reset and start the MPU.

(f) BA - Bus Available - When this signal is active, the MC6802 is stopped and the address bus is available for external devices.

(g) $\overline{\text{HALT}}$ - When this signal is active, all activity of the MC6802 will be halted.

(h) $\overline{\text{IRQ}}$ - Interrupt Request - This signal requests that an interrupt sequence be generated within the MC6802.

(i) $\overline{\text{NMI}}$ - Non-Maskable Interrupt - This signal is similar to $\overline{\text{IRQ}}$ except that the interrupt mask bit in the condition

2.0 Central Processing Unit (cont'd)

code register cannot prevent this interrupt from being executed.

A detailed description of these control signals is given in the appendix. The timing relationship of E, R/$\overline{W}$, and VMA with respect to address and data signals is shown in the following figure.

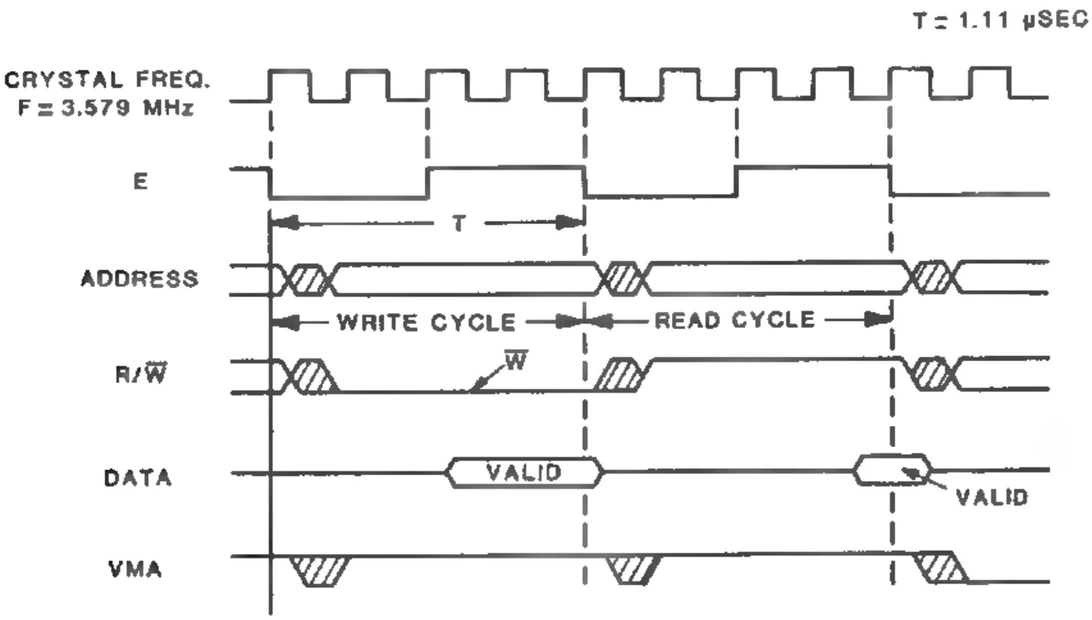


FIGURE 2.0.2. SYSTEM TIMING DIAGRAM

2.1 Memory and Address Decode Logic

As mentioned earlier, the MEK6802D3 has two types of memory devices; RAM and ROM.

RAM: The on board RAM consists of two MCM6810's that contain 128 bytes each of static storage and 128 bytes located within the MC6802 MPU itself. This random access memory is used for the temporary storage of the user program and variable data.

2.1 Memory and Address Decode Logic (cont'd)

The actual location of RAM is determined by the address combination connected to the memory component chip selects. As noted earlier in Figure 1.1.1, the first 128 bytes are located from hex address $0000 to $007F. This memory is contained inside the MPU chip and, as illustrated in the address map of Table 2.1.1, does not require any address decoding. Address lines A0 - A6 select one byte out of the possible 128 bytes. If for some reason the user does not want to use the MPU RAM, it can be disabled by placing a jumper in E7; see system schematic (Appendix 3). A jumper in E7 will ground the RAM enable (RE) input and disable the memory from the bus.

The remaining 256 bytes of RAM are obtained with MCM6810 memory chips. Their address locations are determined by the address combination given in Table 2.1.1. The MCM6810 has six chip selects with only three being used for decode; CS0, CS1, and CS2. One MCM6810 (device U30) has the chip selects set to respond to hex address $0080 to $00FF. However, as noted in Table 1.1.1 and 2.1.1, the address selection can be moved to hex address $8180 to $81FF. This can be accomplished by removing jumpers E4 and E5 and inserting jumpers in E3 and E6.

The other MCM6810 is located at $8100 to $817F. This 128 byte block of RAM is employed as stack memory for the D3BUG monitor and should not be used for general purpose storage of user programs.

ROM: The MEK6802D3 monitor (D3BUG) is contained in the 2K bytes of ROM of the MC6846. In addition to the ROM, the MC6846 has a bidirectional

TABLE 2.1.1. MEK6802D3 ADDRESS MAP

Device	E	R/$\overline{W}$	VMA	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	AØ	ADDRESSES	U#
MCM6810	1	.	1	0	0	0	0	0	0	0	0	1	X	X	X	X	X	X	X	ØØ8Ø - ØØFF	U30
MCM6810	1	.	1	1	0	0	0	0	0	0	1	1	X	X	X	X	X	X	X	818Ø - 81FF	U30
MCM6810	1	.	1	1	0	0	0	0	0	0	1	0	X	X	X	X	X	X	X	81ØØ - 817F	U31
MC68316	1	1	1	1	1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	FØØØ - F7FF	U29
* MC6846	1	1	1	1	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	F8ØØ - FFFF	U9
** MC6846	1	.	1	1	0	0	0	0	0	0	0	1	0	0	0	0	X	X	X	8Ø8Ø - 8Ø87	U9
MC6821	1	.	1	1	0	0	0	0	0	0	0	1	0	0	0	1	0	X	X	8Ø88 - 8Ø8B	U8
*** MC6802	1	.	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	ØØØØ - ØØ7F	U26

X = Don't Care. . = Both

* MC6846 ROM.

** MC6846 I/O + Timer.

*** Inherent to the MC6802 by design; can be disabled for system expansion.

2.1 Memory and Address Decode Logic (cont'd)

I/O port and a timer-counter function. The ROM section of the MC6846 is selected with address select lines MROM and MIO. With these select lines, the ROM is enabled for hex address space $F8ØØ to $FFFF as given in Table 2.1.1. A detailed description of the software is given in Chapter 4 along with an assembly listing (Appendix 1). It was mentioned in Chapter 1 that to enhance the computer board with additional support boards, would require the D3BUG software to be expanded by another 2K bytes. This expansion ROM (MCM68316 - U29) is located in hex address space $FØØØ to $F8FF. The features of this ROM are given in Chapter 3 and a detailed description and the binary code listing are given in the MEK68R2M manual.

ADDRESS SPACE DECODE: The address space of the computer is fully decoded. There are two 74LS138's (one of eight decoder) that decode address lines A7 through A9 and A13 through A15; see system schematic, (Appendix 3). Table 2.1.2 lists the decoded address space. The three most significant address lines, A13 through A15 are decoded into eight 8K block select lines $\overline{\text{ADRØ-1}}$ thru $\overline{\text{ADRE-F}}$. For hex address space $ØØØØ to $1FFF, the decoded select output $\overline{\text{ADRØ-1}}$ will be a logic "Ø". The other decoded lines will be a logic "1". For address space $2ØØØ to $3FFF, select line $\overline{\text{ADR2-3}}$ will be a logic "Ø" and so on for the remainder of the select lines.

Address lines A7 through A9 generate the eight decoded select lines $\overline{\text{DCDØ}}$ thru $\overline{\text{DCD7}}$. Note however from Table 2.1.2 that the decode range

2.1 Memory and Address Decode Logic (cont'd)

for the most significant hex digit is not contiguous. That is, $\overline{\text{DCDØ}}$ is a logic "Ø" for the following address ranges:

(a) \$ØØØØ to \$ØØ7F

(b) \$2ØØØ to \$2Ø7F

(c) \$4ØØØ to \$4Ø7F

(d) \$6ØØØ to \$6Ø7F

(e) \$8ØØØ to \$8Ø7F

(f) \$AØØØ to \$AØ7F

(g) \$CØØØ to \$CØ7F

(h) \$EØØØ to \$EØ7F

The reason for this peculiar address range select is that the enable inputs of the 1 to 8 decoder U11 are driven by address lines A10, A11, and $\overline{\text{A12}}$.

Some of these decoded outputs are logically combined with other address lines or some of the other decode lines, such as $\overline{\text{ADR8-9}}$, to generate the following decode select lines:

(a) I/O1 - \$8ØØØ to \$8Ø7F

(b) I/O2 - \$8Ø8Ø to \$8ØFF

(c) MIO - \$8Ø8Ø to \$8Ø8F

(d) MROM - \$F8ØØ to \$FFFF

The I/O1 and I/O2 lines go to the system bus and define the peripheral I/O address range for an expanded system. See Chapter 3 for additional information on these two signals. The decoded address lines are also used for the decode of RAM and ROM.

TABLE 2.1.2. ADDRESS DECODE MAP

Decode Select	E	R/$\overline{W}$	VMA	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	AØ	ADDRESS SPACE
$\overline{\text{ADRØ-1}}$	X	X	1	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	ØØØØ - 1FFF
$\overline{\text{ADR2-3}}$	X	X	1	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	2ØØØ - 3FFF
$\overline{\text{ADR4-5}}$	X	X	1	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	4ØØØ - 5FFF
$\overline{\text{ADR6-7}}$	X	X	1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	6ØØØ - 7FFF
$\overline{\text{ADR8-9}}$	X	X	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	8ØØØ - 9FFF
$\overline{\text{ADRA-B}}$	X	X	1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	AØØØ - BFFF
$\overline{\text{ADRC-D}}$	X	X	1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	CØØØ - DFFF
$\overline{\text{ADRE-F}}$	X	X	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	EØØØ - FFFF
I/O1	X	X	1	1	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	8ØØØ - 8Ø7F
I/O2	X	X	1	1	0	0	0	0	0	0	0	1	X	X	X	X	X	X	X	8Ø8Ø - 8ØFF
MIO	X	X	1	1	0	0	0	0	0	0	0	1	0	0	0	X	X	X	X	8Ø8Ø - 8Ø8F
MROM	X	X	1	1	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	F8ØØ - FFFF
$\overline{\text{DCDØ}}$	X	X	X	X	X	X	0	0	0	0	0	0	X	X	X	X	X	X	X	*ØØØ - *Ø7F
$\overline{\text{DCD1}}$	X	X	X	X	X	X	0	0	0	0	0	1	X	X	X	X	X	X	X	*Ø8Ø - *ØFF
$\overline{\text{DCD2}}$	X	X	X	X	X	X	0	0	0	0	1	0	X	X	X	X	X	X	X	*1ØØ - *17F
$\overline{\text{DCD3}}$	X	X	X	X	X	X	0	0	0	0	1	1	X	X	X	X	X	X	X	*18Ø - *1FF
$\overline{\text{DCD4}}$	X	X	X	X	X	X	0	0	0	1	0	0	X	X	X	X	X	X	X	*2ØØ - *27F
$\overline{\text{DCD5}}$	X	X	X	X	X	X	0	0	0	1	0	1	X	X	X	X	X	X	X	*28Ø - *2FF
$\overline{\text{DCD6}}$	X	X	X	X	X	X	0	0	0	1	1	0	X	X	X	X	X	X	X	*3ØØ - *37F
$\overline{\text{DCD7}}$	X	X	X	X	X	X	0	0	0	1	1	1	X	X	X	X	X	X	X	*38Ø - *3FF

* = Ø, 2, 4, 6, 8, A, C, E (the most significant hex digit can be any of these values).
X = Don't care.

2.1 Memory and Address Decode Logic (cont'd)

RAM DECODE SELECT

Device	Select	Decoded Address Space
U31	$\overline{\text{DCD2}}$ $\overline{\text{ADR8-9}}$	\$8100 to \$817F
U30 (Two Address Options)	$\overline{\text{DCD1}}$ $\overline{\text{ADR0-1}}$	\$0080 to \$00FF
		or
	$\overline{\text{DCD3}}$ $\overline{\text{ARD8-9}}$	\$8180 to \$81FF

ROM DECODE SELECT

Decive	Select Lines	Address Space
U9	MROM MIO	\$F800 to \$FFFF
U29	$\overline{\text{ADRE-F}}$ $\overline{\text{A12}}$	\$F000 to \$F800

Since the above mentioned decoded address space is used for the control of the on board internal data bus (addresses from \$0100 to \$1FFF are to be included in this address space), this memory space cannot be used off board in an expanded system. To insure this, the bidirectional data buffers (U2 and U3) are forced into a data out mode by signal RDE. The above select lines plus the R/$\overline{\text{W}}$ control and E clock are logically combined through gate U27 to insure that RDE will always force the data buffers into an output mode.

If the \$0000 to \$1FFF address space must be utilized in an expanded system, then the following change must be made to the board:

(a) Remove E9

(b) Insert E10

2.1 Memory and Address Decode Logic (cont'd)

(c) Insert E7

(d) Change the address of U30 from \$ØØ8Ø - \$ØØFF to \$818Ø - \$81FF by changing the jumpers as noted earlier.

For complete jumper details, see Chapter 3.

2.2 I/O

KEYBOARD/DISPLAY: The keyboard and display connect to the on board bus through U8, the peripherial interface adapter (PIA) MC6821. A scanning technique is used on both the keyboad and display to reduce hardware, power, and cost. Since the operation of this circuitry is controlled by the monitor program, refer to the software description in sections 4.2, 4.3, 4.4 and the system schematic (Appendix 3) to follow the description of the keyboard/display operation.

DISPLAY: The scanning procedure uses PIA lines PBØ through PB7 and the seven segment pattern to be displayed is controlled by PIA lines PAØ through PA6. The display routine called PUT (detailed in section 4.2) controls the display logic in the following manner; first the PUT routine determines which of the eight displays will exhibit the desired data by taking one of the PIA PB lines high. For explanation purposes assume PB7 goes high. Next the PUT routine sends the desired seven segement data from DISBUF to the eight displays, U18 through U25. Since PB7 is high, the selected segments of U18 will be turned on. PBØ through PB6 remain low preventing U19 through U25 from displaying data. Then the PUT routine delays for one millisecond. During this one millisecond, U18 will be

2.2 I/O (cont'd)

turned on. Next the PUT routine will select the data pattern to be displayed on U19, which is the next display and PIA bit PB6 will be set and PB7 cleared. This sequence continues until the right most digit, U25, has been selected. Next the program returns to display U18 and the sequence repeats itself. This sequence or refresh rate is sufficient to make the displays appear to be on continuously.

Note that (system schematic, Appendix 3) the PIA lines PBØ through PB7 and PAØ through PA6 are buffered. The buffers U17, U14, U15, U33 and U34 are necessary to supply the high current required by the LED displays.

KEYPAD: The keypad has 25 keys that are electrically arranged in four columns by six rows. The reset key (RS) is not decoded and is not part of the row/column matrix; see Appendix 3, sheet 3. Note that the rows of the key matrix are driven by the select display driver buffers U19, U33, and U34. Thus as the PUT routing sequentially selects the displays U20 through U25, the corresponding rows of the key matrix will be selected. The select state is a logic "Ø" or ground. The four columns of the matrix are connected to NAND gate U26B and the four to one demultiplexer, U28A.

When one of the keys in the matrix is depressed, the output of gate U27B will be forced to a logic "1" when the row buffer that the key is attached to goes low. The logic "1" from U27B causes the PIA U8 to set an $\overline{\text{NMI}}$ interrupt, see section 4.3 for a description of the Keypad/Decode routine. When the keypad interrupt is acknowledged, all of the displays are blanked by storing ØØ to the segment or data lines of the displays. Next a routine

2.2 I/O (cont'd)

is used to determine the column of the key depressed. This routine sequences PIA lines PB6 and PB7. When the output of the four to one demultiplexer goes low, the PIA PA7 line is driven low. This low on PA7 will tell the routine which column was selected by the binary state of the PB6 and PB7 lines.

Next a row find routine is executed. This routine sequentially drives each one of the PBØ through PB5 lines high until the active row is found. There is a key table that corresponds to the key matrix, so when the row and column address of the depressed key is found, the key table can be accessed and the value of the key loaded into the A accumulator. The reset button does not require decoding because it is tied to the power up reset logic which drives the $\overline{\text{RESET}}$ input of the 6802 directly.

SK1 SOCKET: The MEK6802D3 board has one I/O port available for the user other than the D3 I/O system expansion bus. A 16 pin socket SK1 allows the user to interface to the timer and parallel port of the MC6846. The parallel bidirectional I/O port has functional operational characteristics similar to the B port on the MC6821 PIA. These include 8 bidirectional data lines, PØ through P7, and two handshake lines, CP1 and CP2. The control and operation of these lines are completely software programmable. The PIA timer address space is from \$8Ø8Ø to \$8Ø87, see Figure 1.1.1.

The timer-counter port is a 16 bit binary counter which under software control may be programmed to count events or measure frequencies and time intervals. It can also be used for square wave generation, single pulses

2.2 I/O (cont'd)

of controlled duration, and gated signal interrupts may be generated from a number of conditions selectable by software. The CTO (counter timer output) signal is used by D3BUG software to generate a $\overline{NMI}$ during single step operation.

CHAPTER 3

EXPANSION DESCRIPTION

3.0 Introduction

For those who are interested in having a more versatile and powerful computer, the MEK6802D3 can be easily expanded through the system bus. The D3 can interface directly to the following support cards, which are referred to as the MOKEP series boards:

(1) MEK68R2M Programmable CRT Interface.

Description:

Software programmable line and character format.

a) 16 x 32, 16 x 64, or 20 x 80.

b) User defined format.

Display features.

a) 5 x 7 matrix.

b) Upper and lower case, full 128 ASCII character set.

c) Semi-graphic capability with up to 160 x 72 screen resolution and two levels of gray shading.

d) Complete cursor control.

Memory.

a) MCM68316E (2K pre-programmed ROM) included to expand MEK6802D3 to full 4K operating system.

b) Designed to accept up to 4K bytes of RAM (eight MCM2114's as screen refresh RAM.

c) Module supplied with 1K byte of MCM2114 RAM.

d) System MPU has access to display memory.

3.0 Introduction (cont'd)

Interfaces to:

a) CRT monitor, or

b) Modified TV set.

c) 128 character ASCII encoded keyboard.

d) MEK6802D3

Minimal Power Requirements.

a) Single 5 volt supply at 1.1 Adc max.

b) ± 12 volt from bus routed to keyboard interface socket.

c) Provision for on board negative 5 volt regulation for keyboard interface socket.

(2) MEK68IO Input/Output Module.

Description:

Parallel input/output.

a) Provisions for two MC6821 PIA's gives total of four 8 bit ports with eight control lines.

b) One MC6821 furnished with module.

Audio cassette interface.

a) 300 baud Kansas City Standard.

b) 1200 baud Modified Kansas City Standard.

c) Both 300 and 1200 baud fully populated and supported by MEK6802D3 firmware.

Serial input/output.

a) MC14411 baud rate generator provides all standard communications rates.

3.0 Introduction (cont'd)

b) MC6850 ACIA with both RS232C and 20 mA current loop interface.

c) Optically isolated port termination.

IEEE 488 interface.

a) Provisions for user installation of MC68488 and GPIB buffers.

Power requirements.

a) $\pm$12 volt utilized by serial I/O only.

b) +5 volt required for all sections.

(3) MEK68MM Dynamic Memory Module.

Description:

Hidden refresh operation.

Available in two configurations.

a) 32K byte as MEK68MM32.

b) 16K byte as MEK68MM16.

Low power consumption.

a) VCC (+5 V) requires 820 mA for either 16K bytes or 32K bytes.

b) $\pm$ 12 volts.

RAM paging feature fully supported.

(4) MEK6802RR ROM/RAM Module.

Description:

Contains two blocks of four ROM/EPROM sockets each.

a) Either block can utilize 1K, 2K, 4K, or 8K byte ROM/EPROM.

b) ROM/EPROM with single or multiple supplies can be used.

c) ROM paging feature supported by each block.

Contains two blocks of eight MCM2114 sockets each.

3.0 Introduction (cont'd)

a) Either block is independently relocatable to any even 4K starting location (XØØØ, where X = any hex number from Ø - F).

b) RAM paging feature supported by each block.

c) Each block has Write disable switch to allow ROM simulation during debug.

d) Two MCM2114's (1K byte) supplied with module.

Power requirements.

a) Option on board negative 5 volt regulator for use with EPROM's.

b) Population options determine current and voltage requirements.

(5) MEK68VG Color TV Monitor Interface.

Operating modes.

a) System is operable in alphanumeric, semi-graphic, and full graphic modes.

b) Internal ROM creates ASCII characters for alphanumeric mode.

c) Alphanumeric mode displays 32 x 16 character density and has selectable video inverse.

d) Semi-graphic modes offer 8 colors in 64 x 32 density or 4 colors in 64 x 48 density.

e) Full graphic mode offers 2 or 4 colors and 64 x 64, 128 x 64, 128 x 96, 128 x 192 or 256 x 192 density.

Memory.

a) MCM68316E (2K pre-programmed ROM) included to expand MEK6802D3 to full 4K operating system.

b) Designed to accept up to 6K bytes of RAM (twelve MCM2114's).

c) Module supplied with 1K byte of MCM2114 RAM.

d) Dual density addressing allows 6K of display with 1K of memory.

e) System MPU has access to display memory.

Interfaces to:

a) Color TV Monitor, or

b) Modified color TV set.

c) 64 character ASCII encoded keyboard.

Single 5 volt supply.

(6) MEK68WW Wire Wrap/Extender Card.

Description:

Supports standard 0.3" and 0.6" wide DIP's.

a) 26 columns with 0.3" spacing.

b) Each column contains 42 holes with 0.1" spacing.

c) Provisions for two decoupling capacitors per column.

Bus interface.

a) Address, Control and Data Bus buffers supplied with board for user installation.

b) Seventy-pin female connector allows ten user defined signals to be routed to motherboard. (MEK68MB utilizes 70 pin male connector).

Male right angle connector (60 or 70 pin).

a) Allows use as extender card.

b) Can be used as cable connector.

Unassembled kit allows user to define configuration.

3.0 Introduction (cont'd)

(7) MEK68EP EPROM Programmer Module.

Description:

Programs wide variety of products.

a) 1K, 2K or 4K byte EPROM's.

b) Single 5 volt or multiple supply versions.

c) Motorola, Intel or TI pinouts.

EPROM user sockets.

a) After programming, up to four EPROM's (or ROM's) can be inserted in on board sockets for system usage.

b) Sockets fully support ROM paging feature.

c) Jumper options allow location in User-Defined Address Space.

Programming features.

a) On board DC to DC converter supplies programming voltage.

b) Firmware provided contains programming routines.

c) Copy and verify functions incorporated.

Two-board construction.

a) All electronic circuitry contained on standard 7.0" x 8.25" board.

b) Separate EPROM programming socket module plugs into top of main board, or - with User Supplied Cable - can be remotely located.

c) EPROM programming socket contains Easy In - Easy Out socket.

Power supply requirements.

a) +5 volts utilized by all components.

b) ± 12 volts required if multiple supply EPROM's used.

3.0 Introduction (cont'd)

(8) MEK68CC Card Cage for use with MEK68MB.

All aluminum construction.

Easy 15-minute assembly.

Supplied with connectors to fully expand MEK68MB.

Dimensions of 8-1/4" high x 7-1/4" wide x 13-1/4 deep.

a) Fits one-half of standard cabinet such as Zero Manufacturing P/N 170822.

b) Other half free for power supply, mini-floppy, etc.

c) Extra length provides for mounting of cooling fan or power supply behind motherboard.

d) Designed for convection or forced air cooling.

(9) MEK68MB Motherboard for MOKEP Series.

Description:

Seventy-pin connectors.

a) Accept all MOKEP series.

b) Provide for expansion.

Five slots populated.

a) Stand-alone card guides permit use without Card Cage.

b) 1-1/4" spacing between populated slots.

c) Five additional slots for subsequent upgrade with Card Cage.

Rugged construction.

a) 0.062" thick epoxy G-10 material.

b) Aluminum board stiffeners serve as mounting base.

c) Terminal block for power supply connections.

3.0 Introduction (cont'd)

(10) MK68CMB Card Cage and fully populated (10 slot) motherboard.

Description:

Same features as MEK68CC and MEK68MB.

3.1 System Configuration.

To expand from the single board computer, the user has several options depending on the use and application. The following examples illustrate some of the possible system configurations; from the small dedicated system to the large and flexible type.

(1) The MEK6802D3 plus:

(a) MEK68IO

(b) MEK68MB

This configuration will allow the user to write programs up to 256 bytes and provide permanent program storage through an audio cassette. With the MEK68WW board, this system could be used to control a variety special or unique I/O devices.

(2) The MEK6802D3 plus:

(a) MEK68IO

(b) MEK68MB

(c) MEK68RR

The addition of the MEK68RR can increase the users dynamic programming storage space from 256 bytes to 8,448 bytes. The MEK68RR card also provides the user with ROM storage space that will permit conversion of often used software to permanent storage or firmware.

3.1 System Configurations (cont'd)

(3) The MEK6802D3 plus:

(a) MEK6810

(b) MEK68MB

(c) MEK68MM

(d) MEK68R2M

A system configured with these boards will also require an ASCII keyboard and TV or monitor. This is a minimal system configuration for those users desiring to write and develop sophisticated software. If colored graphics were desired, then the MEK68VG board would be used instead of the MEK68R2M. Also additional memory cards can be added to accommodate larger software requirements.

For special or custom interface requirements, the MEK68WW can be used. Also, for those applications requiring an enclosure, the all aluminum card cage (MEK68CC) can be employed. This card cage was designed to fit into one-half of a standard cabinet such as Zero Mfg., P/N 178022.

3.2 Expanded Memory Address Space.

As mentioned earlier, provisions were made for system expansion to accomodate multi-level memory paging. This is accomplished by using the 8 bit bidirectional data port of the MC6846. Six of the possible eight bits are used: three of the data bits PØ through P2 are buffered (ROPØ - ROP2) and used on the system bus to select one of eight ROM pages, see Figure 3.2.1. The user address space for the eight ROM pages is 20K bytes each.

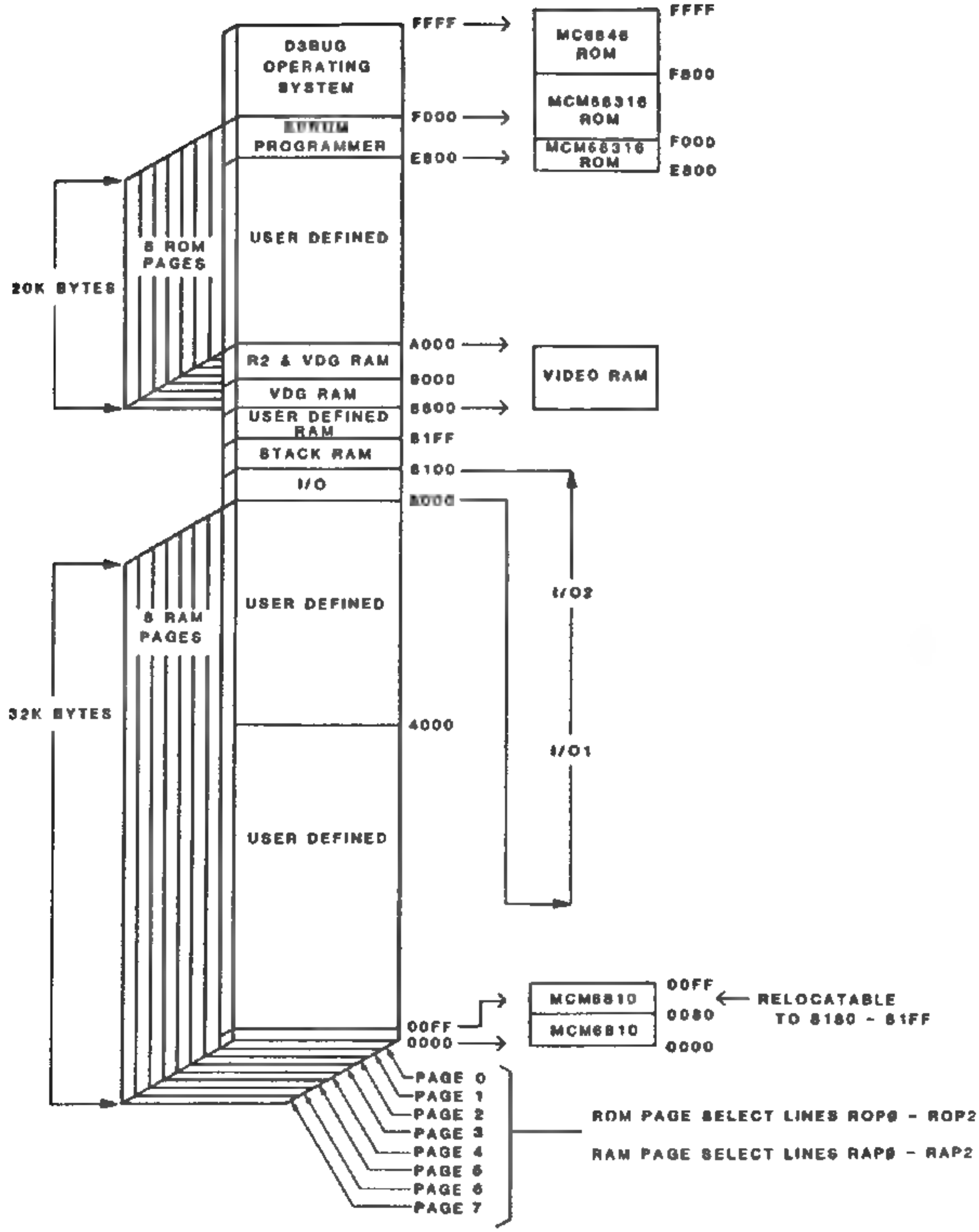


FIGURE 3.2.1. EXPANDED MEK6802D3 MEMORY MAP

3.2 Expanded Memory Address Space (cont'd)

The remaining 3 bits, P3 through P5, are also buffered and are used on the system bus for selection of one of eight RAM pages, see Figure 3.2.1. The user address space for eight RAM pages is 32K bytes each. The user has control of the page addresses through software control of the MC6846 PIA port. For detailed information on use and control of the PIA port, consult the MC6846 data sheet in the appendix.

3.2.1 Expansion Firmware Features.

In Chapter 1, (paragraph 1.5) it was mentioned that the D3BUG contained a punch and load function for an audio cassette. To use the following described punch and load routine requires that an MEK68IO board be on the system bus.

P/L: The Punch/Load key is used to punch/load/verify cassette tapes in two different formats and at two different baud rates. One is the 300 baud Kansas City standard and the other is a 1200 baud binary format with check sum. The baud rate is selected on the I/O card by the user.

To punch a tape switch the baud rate selector on the I/O board to either 300 or 1200 baud. Type P/L and the display will prompt with a 'B' for beginning address. Enter the address and type GO. The display will then prompt with an 'E' for ending address. Enter the address and either type GO, for a 1200 baud tape or type FS and then GO for a 300 baud. When finished the display returns with a 'B'. After punching the tape you may verify it by rewinding the tape and either type M to verify a 1200 baud tape or type FS and then M to verify a 300 baud tape. If the tape is

3.2.1 Expansion Firmware Features (cont'd)

good, the display returns with a 'B' and the ending address that was entered earlier. If the verify fails, the display returns with a 'B' and the address of the bad byte.

To load a tape, type FS and then P/L. A display '12 3 ?' will appear prompting for either a 1200 baud formatted tape (12) or a 300 baud formatted tape (3). The numbers refer to the baud rate at which the two formats are recorded. To load a 1200 baud tape, place the recorder in its play mode and type GO. To load a 300 baud tape, type FS and then GO.

You may also verify a tape in the Load mode in the same manner described in the Punch mode. However, when the verify fails the display 'CS ?' appears. If the verify passes, then '12 3 ?' appears.

EXPANSION ROM: To accommodate other expansion boards, the D3BUG monitor was enhanced by an additional 2K bytes of code. The 2K expansion ROM, device U29, for the MEK68R2M contains the following features:

(1) Memory display/change with offset calculation.

(2) Register, page display/change.

(3) Breakpoint display/change/delete.

(4) Trace single/multiple instructions, display registers, stack pointer, program counter, and up to four 16 bit locations selected by the user.

(5) GO TO user program.

(6) Continue execution.

3.2.1 Expansion Firmware Features (cont'd)

(7) Punch/load, verify, offset/load 300 or 1200 baud.

(8) Examine block of memory with ASCII equivalent.

(9) User defined functions (3).

(10) RS232 handler.

(11) Allow paging control.

For details on these additional features, consult the MEK68R2M user manual. The expansion ROM goes in socket U29 on the MEK6802D3 board.

I/O ADDRESS MAP: The address space for the various expansion boards is given in the expanded memory map of Figure 3.2.2. A detailed control and address map for the boards is illustrated in Table 3.2.1. Note that for I/O address continuity, the I/O and timer of the MC6846 and the keypad/display PIA of the MEK6802D3 computer board were included.

3.3 I/O Expansion Bus.

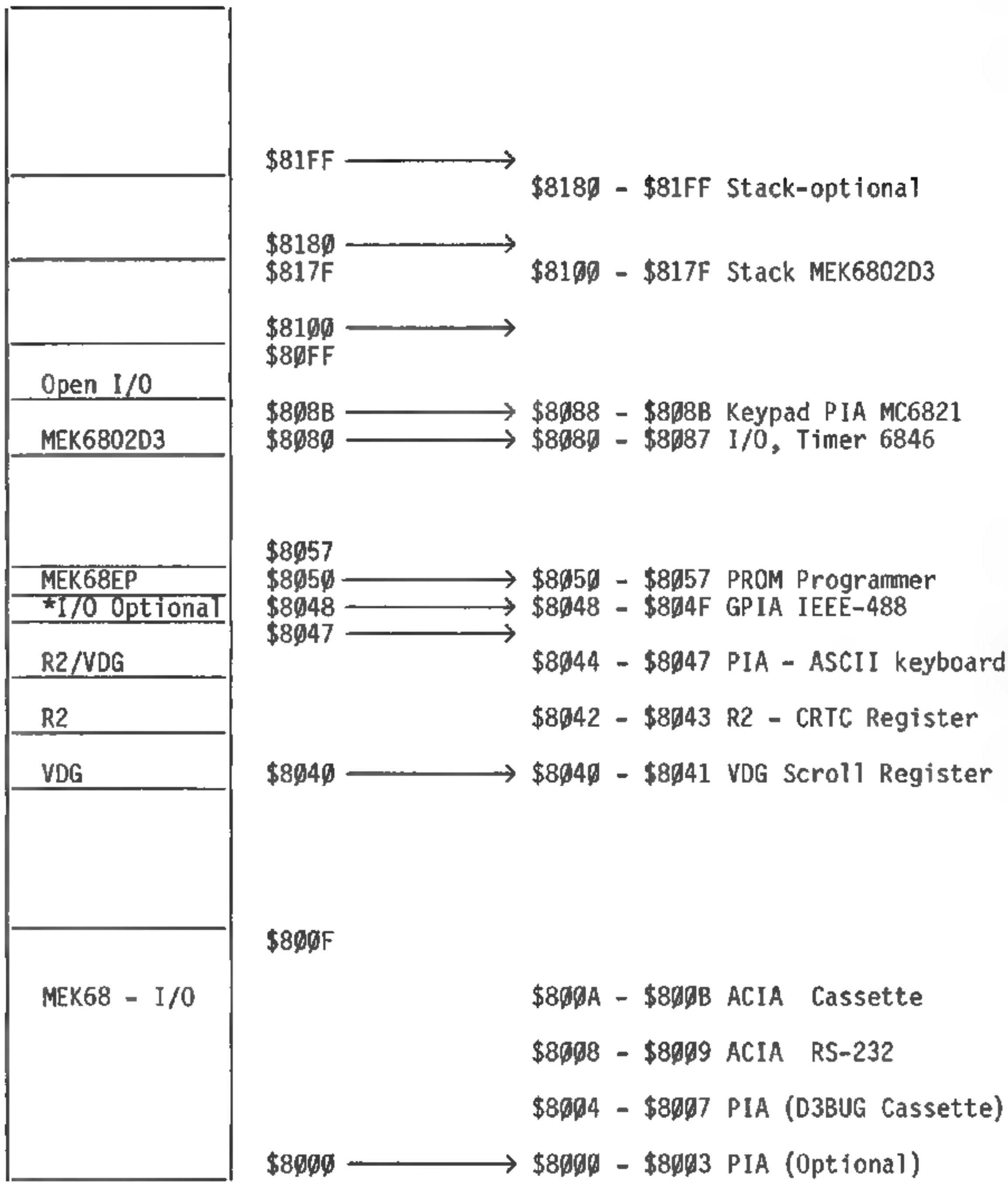
SYSTEM BUS: The external system bus is used to interconnect the various expansion boards to the MEK6802D3 computer board. Table 3.3.1 lists the pinouts of all the signals on this bus.

The external bus contains all of the internal bus as described in section 2.0 plus the following:

(1) 3 - ROM page select lines (ROPØ through ROP2).

These lines are used to select one of eight possible ROM pages and their logic state is controlled through the user's program; see Figure 3.2.1.

I/O MEMORY MAP: MOKEP



*See MEK68IO manual for details.

FIGURE 3.2.2. EXPANDED I/O MEMORY MAP

TABLE 3.2.1. I/O CONTROL AND ADDRESS MAP

Selected Function User PIA	VMA	E	I/01	I/02	A6	A5	A4	A3	A2	A1	A0	Address	Board
User PIA	1	1	1	0	0	0	0	0	0	X	X	8000 - 8003	MEK68IO
Cassette PIA	1	1	1	0	0	0	0	0	1	X	X	8004 - 8007	MEK68IO
ACIA RS-232	1	1	1	0	0	0	0	1	0	0	X	8008 - 8009	MEK68IO
ACIA Cassette	1	1	1	0	0	0	0	1	0	1	X	800A - 800B	MEK68IO
*IEEE - 488 GPIA	1	1	1	0	1	0	0	1	X	X	X	8048 - 804F	MEK68IO
VG Scroll Register	1	X	1	0	1	0	0	0	0	0	X	8040 - 8041	MEK68VG
R2 - CRTC Register	1	1	1	0	1	0	0	0	0	1	X	8042 - 8043	MEK68R2M
ASCII Key Board/PIA	1	1	1	0	1	0	0	0	1	X	X	8044 - 8047	MEK68R2M
PROM Programmer	1	1	1	0	1	0	1	0	X	X	X	8050 - 8057	MEK68EP
I/O + Timer	1	1	0	1	0	0	0	0	X	X	X	8080 - 8087	MEK6802D3
Keypad/Display PIA	1	1	0	1	0	0	0	1	0	X	X	8088 - 808B	MEK6802D3

*Circuitry on board only - user supplied parts.

TABLE 3.3.1 MEK6802D3 SYSTEM BUS PIN OUTS

Pin Number	Signal	Pin Number	Signal
1	GND	31	A1
2	E (Ø2)	32	A2
3	+5 V STBY	33	A3
4	+5 V	34	A4
5	OPEN	35	A5
6	OPEN	36	A6
7	OPEN	37	A7
8	MR	38	A8
9	VMA	39	A9
10	BA	40	A10
11	R/$\overline{W}$	41	A11
12	GND	42	A12
13	$\overline{RESET}$	43	A13
14	$\overline{NMI}$	44	A14
15	$\overline{IRQ}$	45	A15
16	$\overline{HALT}$	46	GND
17	OPEN	47	GND
18	OPEN	48	DØ
19	GND	49	D1
20	KEY SLOT	50	D2
21	ROPØ	51	D3
22	ROP1	52	D4
23	ROP2	53	D5
24	RAPØ	54	D6
25	RAP1	55	D7
26	RAP2	56	+12 V
27	I/Ø1	57	GND
28	I/Ø2	58	-12 V
29	OPEN	59	+5 V
30	AØ	60	GND

3.3 I/O Expansion Bus (cont'd)

(2) 3 - RAM page select lines (RAPØ through RAP2).

These lines are used to select one of eight possible RAM pages and are also controlled through the user's software; see Figure 3.2.1.

(3) 2 - I/O select lines.

These decoded lines I/01 and I/02 were described in section 2.1. I/01 select line is a logic "1" for address space 8ØØØ to 8Ø7F and a logic "Ø" for all other address combinations.

I/02 select line is a logic "1" for address space 8Ø8Ø to 8ØFF and a logic "Ø" for all other address combinations. Both the D3BUG monitor and the user's program control the logic state of these lines.

BUS CONTROL: As noted earlier in section 2.0, the MR line is used to allow the MEK6802D3 card to interface with cards that have cycle times slower than the D3. This mode of operation is accomplished by pulling the MR line low, logic "Ø". When MR is low, clock signal E will be stretched integral multiples of half periods, as noted in Figure 3.3.1, thus allowing interface to slow cards.

BUS ELECTRICAL CHARACTERISTICS: Refer to Table 3.3.2 for the system bus electrical characteristics. The MEK6802D3 can accommodate up to ten standard expansion boards.

JUMPER OPTIONS: The MEK6802D3 board contains numerous jumpers which allow it a great deal of flexibility during system expansion. These

3.3 I/O Expansion Bus (cont'd)

TABLE 3.3.2. BUS ELECTRICAL CHARACTERISTICS

Signals	Parameter	Min	Max	Unit (dc)
AØ - A15, ROPØ - ROP2, RAPØ - RAP2, E, VMA, BA, R/$\overline{W}$	VOL @ IOL = +24 mA		0.5	V
	VOH @ IOH = -15 mA	2.4		V
DØ - D7	VOL @ IOL = +24 mA		0.5	V
	VOH @ IOH = -15 mA	2.4		V
	IIH @ VIH = 2.7 V		0.2	mA
	IIL @ VIL = 0.4 V		-1.6	mA
$\overline{IRQ}$, $\overline{HALT}$, $\overline{NMI}$, $\overline{RESET}$, MR	IIH @ VIH = 2.7 V	-7.0		mA
	IIL @ VIL = 0.4 V		-1.6	mA
I/01, I/02	VOL @ IOL = +12 mA		0.4	V
	VOH @ IOH = -1.2 mA	2.4		V

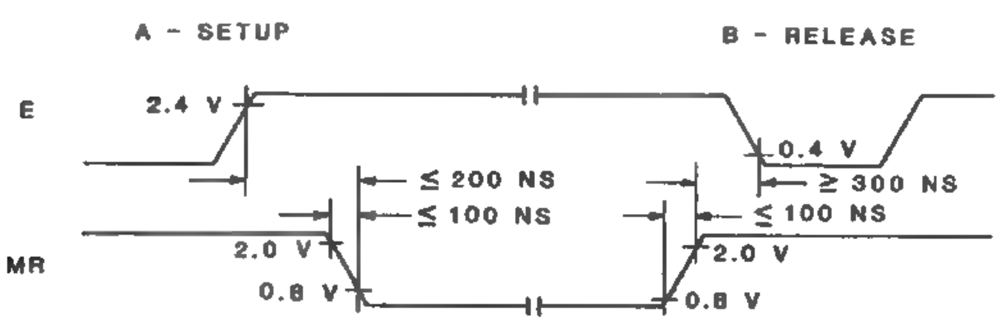


FIGURE 3.3.1. MEMORY READY CONTROL FUNCTION

jumpers are actually zero ohm resistors and are given 'E' numbers to denote their locations.

(A) Jumpers E1 and E2 are used to extend the address range of the MOKEP operating system. The board is configured with E1 removed and E2 installed. This locates signal 'ADDR-F' from \$F000 to \$FFFF. In the other configuration, 'ADDR-F' is extended to \$E000 to \$FFFF. This affects the off board Read Enable 'RDE' signal locking out \$E000 to \$FFFF from being off board. This allows the 'MROM' signal to occupy two locations in the MOKEP Memory Map, \$E800 to \$EFFF and \$F800 to \$FFFF simultaneously.

(B) Jumper pairs E4-E5 and E3-E6 are used to relocate the MC6810 from \$0080 - \$00FF to \$8180 - \$81FF for use as an auxillary stack. The board is configured with E4 and E5 installed. To reconfigure it, jumpers E3 and E6 should be installed instead. Do not install any other combination, as they will cause the RAM to conflict with other components in the memory map.

(C) Jumper E7 is used to disable the 128 bytes of RAM located in the MC6802. The board is configured with E7 removed. During system expansion, install E7 which will force RAM enable (RE) low disabling the RAM.

(D) Jumper E8 does not exist.

3.3 I/O Expansion Bus (cont'd)

(E) Jumpers E9-E10 are used to enable or disable reading off board from \$ØØØØ ⟶ \$1FFF. The board is configured with E9 installed and E10 removed during system expansion E10 is install and E9 is removed.

JUMPER OPTIONS: SUMMARY.

Single Board		Full System Expansion (Note 1)	
E1 :	Removed	E1 :	Removed
E2 :	Installed	E2 :	Installed
E3 :	Removed	E3 :	Installed
E4 :	Installed	E4 :	Removed
E5 :	Installed	E5 :	Removed
E6 :	Removed	E6 :	Installed
E7 :	Removed	E7 :	Installed
E9 :	Installed	E9 :	Removed
E10 :	Removed	E10 :	Installed (Note 2)

Note 1 : Including off board RAM module.

Note 2 : A 10K ohm resistor can be used.

CHAPTER 4

SOFTWARE DESCRIPTION (D3BUG MONITOR)

4.0 General Description

D3BUG, a 2K x 8 bit monitor program, is provided with the MEK6802D3. It resides in the MC6846, which is a combination ROM, timer and interface port. This chapter provides a description of data storage locations, flow charts of each major section of D3BUG, and a detailed explanation of how to use software routines residing in D3BUG in your own programs. In addition, an assembly listing, complete with a cross-reference symbol listing, is provided in the appendix.

D3BUG requires 128 bytes of RAM memory for its operating stack. This is located from $8100 to $817F. This stack RAM is used as temporary data storage for the 6802 processor when a stack operation is required such as a push or pull instruction or when a subroutine or interrupt is encountered.

Several RAM locations are used for temporary data storage and as flags by the monitor for communicating between various routines. Some of the more significant locations are described below and are referred to in the description of D3BUG.

MNPTR ($8100)	Contains the address of program to be executed during the PUT routine.
UHASH ($8102)	User pointer to special function look up.
UNMIV ($8104)	User non-maskable interrupt vector - store user NMI program address here.
UIRQV ($8106)	User interrupt request vector.

4.0 General Description (cont'd)

USWIV ($81Ø8)	User software interrupt vector.
DIGEN ($8115)	Rotating digit enable bit for display.
KEY ($8117)	Decoded Key value from keypad interrupt.
HEXBUF ($8118)	4-byte hex buffer for the LED display - 2 digits per byte.
DISBUF ($811C)	8-byte display buffer (7-segment coded).
USP ($812B)	User stack pointer.
ROWCOL ($812E)	Contains the coded key closure derived from the GET routine interrogating a keypad interrupt.
ROLFLG ($8134)	Flag used by the ROLENT subroutine to indicate to the routine that leading zeros are required when set to zero. If ROLFLG is set to a one, the routine will shift the currently displayed numbers to the left and store the new value contained in the A accumulator in the rightmost digit.
FLG24 ($8135)	Flag used by the ROLENT subroutine to indicate which displays to update. If FLG24 is set to zero, the update of displays 1-4 is selected; if set to a one the update of display 5-6 is selected.
TOFT ($814Ø)	A 2-byte pointer is used by the breakpoint subroutine that holds ($814Ø) the current pointer address of the breakpoint table.

4.1 Restart/Initialization Routine

The restart/initialization routine is included in the flow chart in Figure 4.1.1. When the RESET (RS) key is released, the 6802 processor outputs addresses $FFFE and $FFFF which are the processor's restart vector address locations and contain the program address of RESTRT. This initialization program is located at $F8ØØ, the beginning location of the D3BUG ROM.

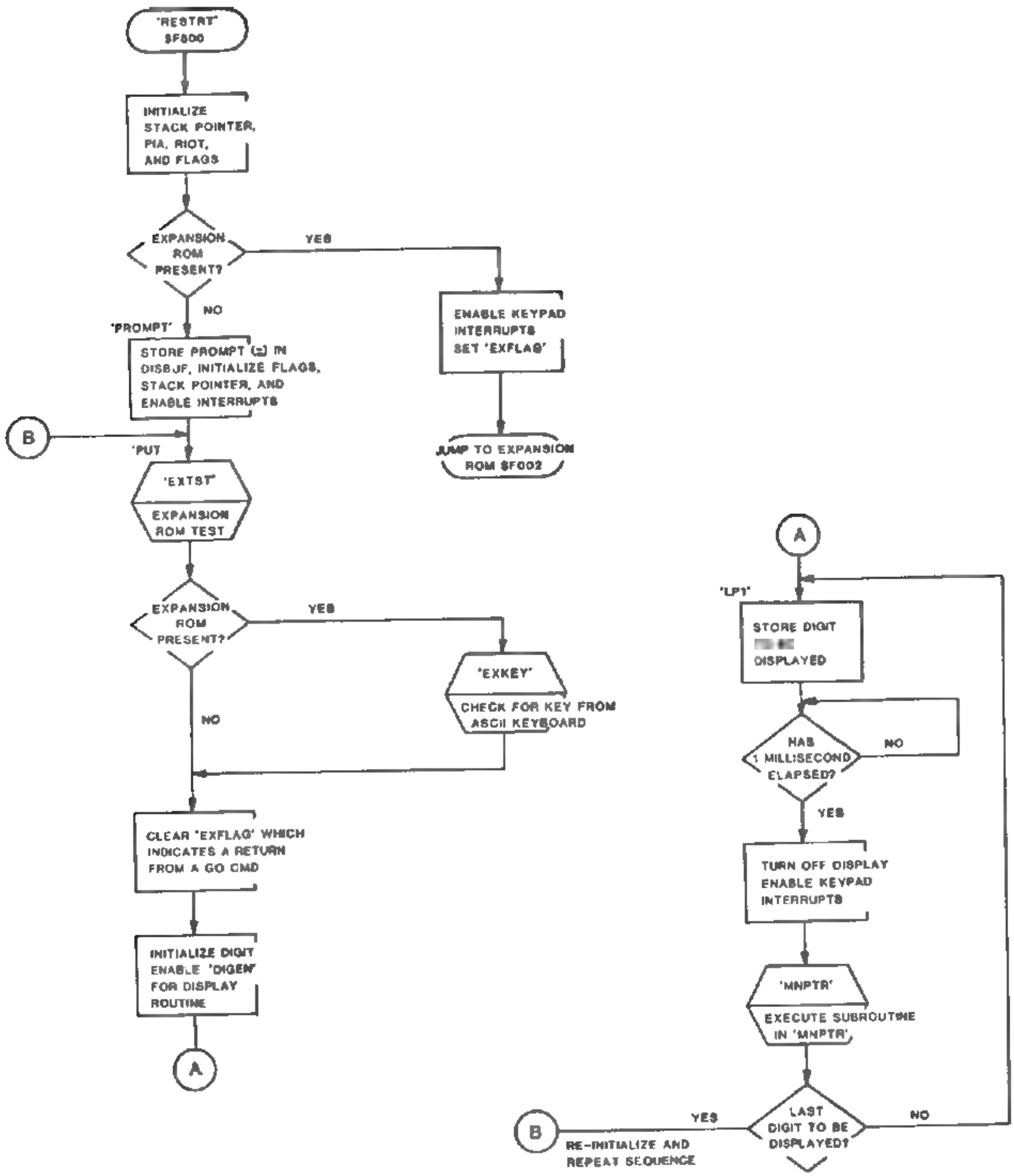


FIGURE 4.1.1. OVERALL PROGRAM FLOW FOR D3BUG MONITOR

4.1 Restart/Initialization Routine (cont'd)

RESTRT initializes the stack pointer to $817F and sets the interrupt mask to protect against interrupts occuring during initialization. The user stack pointer is initialized to a default address of $ØØFF, which is the top of the user RAM supplied with the D3. If it is desired not to relocate the user stack pointer to other than the default address of $ØØFF, then do not write programs that would reside in that area of memory.

The next step is the initialization of the timer which is used by the trace command. The fresh start flag (INIT1) is cleared and the top of the table called TOFT is initialized with a $ØØØF which is used in the break-point routine (D3BREAK). The display/keypad PIA, the MC6821 located at $8Ø8Ø-$8Ø83; and the memory paging control interface, part of the MC6846 located at $8Ø88-$8Ø8A are initialized. A subroutine called DISNMI is called to disable the $\overline{\text{NMI}}$ trace timer in order to prevent accidential triggering during initialization.

The program then executes a subroutine that interrogates the presence of an expansion ROM located at $FØØØ. If the ROM is present, the subroutine that enables keypad interrupts (EFPI) is called and EXFLAG is set to indicate that the expansion ROM is present. The program then jumps to the expansion ROM entry point $FØØ2. If no expansion ROM is detected, the program exits to the PROMPT routine which continues the initialization sequence.

The subroutine called CLRD clears the 8 LED displays and then the program resets the keypad interrupt flag and displays an equal sign prompt (=)

4.1 Restart/Initialization Routine (cont'd)

on the leftmost display. ROWCOL is initialized to $8Ø to indicate there is no key pending and clears other miscellaneous flags. Next the D3BUG stack pointer is again reset to $817F. Main pointer (MNPTR) is initialized with the address of the D3BUG function decode routine. The subroutine called ENNMI is executed to enable interrupts and control is passed to the routine called PUT.

4.2 Display/Command Execution Routine

The display routine (PUT) is detailed in Figure 4.2.1 and is located at address $F9ØB in D3BUG. The PUT routine performs 3 tasks. It is responsible for both updating and multiplexing the LED displays and executing a command subroutine defined by the memory location MNPTR.

The PUT routine begins by checking for the presence of the expansion ROM. If the ROM is detected, it then checks for a key closure from an ASCII keyboard. If a key closure is found, control is turned over to the expansion ROM; otherwise, it returns to the main program flow.

Next the flag called EXFLAG is reset. DIGEN, a temporary byte location used to control the digit enable for the display circuit, is initialized. The next sections of PUT are LP1 and LP2, which are program loops designed to output the contents of the display buffer (DISBUF) to the D3 display LEDs. The display buffer is a temporary storage area of RAM that contains the data for the eight seven-segment displays. LP1 begins by storing DIGEN to the PIA controlling the display. The X register is initialized to point

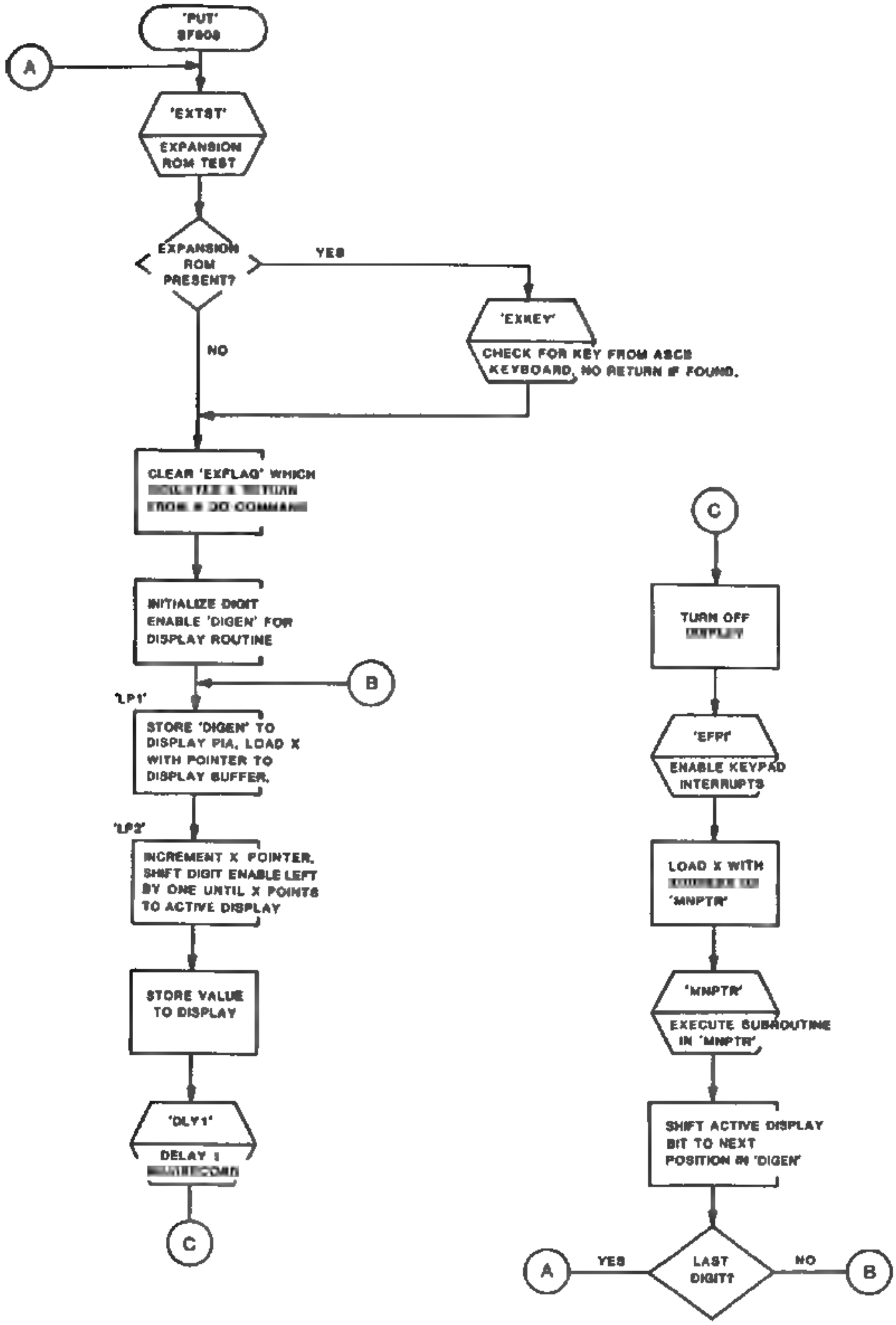


FIGURE 4.2.1. DISPLAY/COMMAND EXECUTION

4.2 Display/Command Execution Routine (cont'd)

to the beginning of the display buffer. LP2 is the internal loop which increments the X pointer by one, until the carry bit is set. The X register is now pointing at the active display. The routine stores the value from the display buffer to the display through the PIA. At this point the program delays for 1 millisecond. This allows the display to be turned on for the length of the delay before blanking and going on to update the next display. The keypad interrupts are enabled through the subroutine called EFPI insuring that a key depression will be recognized. The next step executes the subroutine stored in MNPTR. During restart this location is initialized to point to the function decode routine which detects a key closure, determines the key function and stores the address of the subroutine corresponding to the key function in MNPTR.

The program next shifts the digit select (DIGEN) to the next display position and goes back to LP1 to repeat the process for all eight display positions. When completed, the program returns to the beginning of the PUT routine.

4.3 Keypad/Decode Routine

After receiving a keypad $\overline{\text{NMI}}$ interrupt, the NONMSK routine determines that it is a keypad interrupt and executes the GET subroutine to determine which key was depressed. The GET routine first tests the expansion ROM flag EXFLAG. If an expansion ROM is present, the program exits to the expansion ROM routine called EXGET. If not, the routine blanks all displays by storing a $ØØ to the PIA. The COLUMN routine tests each column

4.3 Keypad/Decode Routine (cont'd)

to determine which key is depressed and, when found, the column number is stored in the MSB's of the A accumulator.

The COLFND routine saves the column number and then stores a one in the A accumulator. The ROW routine begins testing each row, shifting the one bit in the A accumulator left by one and looping back through ROW until the active row is found. The ROWFND routine stores the ROW information into the 4 LSB's of the A accumulator, combines the column with the row found, and stores the result in ROWCOL for later. The CLOP routine checks for key release. It continues in a loop until the key is released. The KEYCOD routine massages the row and column data again, this time to form a table offset. The address of the key table is loaded into the X register and the subroutine ADDXA is executed to force the effective table address. The key value is loaded into the A accumulator and saved in KEY for later. The KEYCOD routine then delays for 25 milliseconds. The $\overline{\text{NMI}}$ keypad interrupt is then reset and control returns to the interrupted program.

4.4 Function Decode Routine

The function decode routine (FNCDCD) is used by D3BUG to determine the function of a key depressed. The routine first checks for a key closure. If none is found, FNCDCD returns to PUT. If a key is found then FNCDCD continues with the key value in the A accumulator. If the key is a function key, the appropriate routine corresponding to it is found in a look-up table and the address is stored in MNPTR to be executed later in the PUT routine. The program then returns to the caller. If a hex key is found,

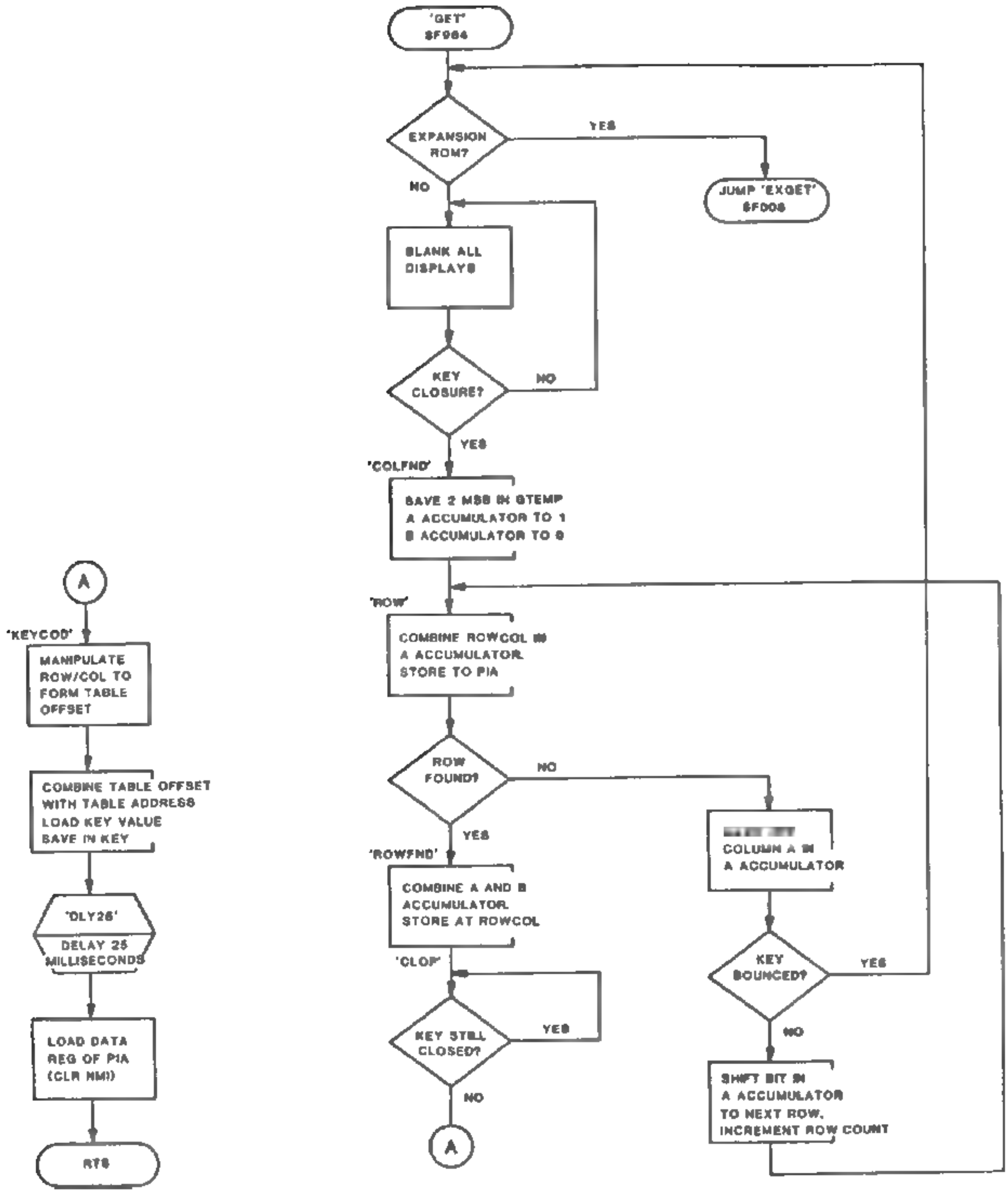


FIGURE 4.3.1. KEYPAD DECODE ROUTINE

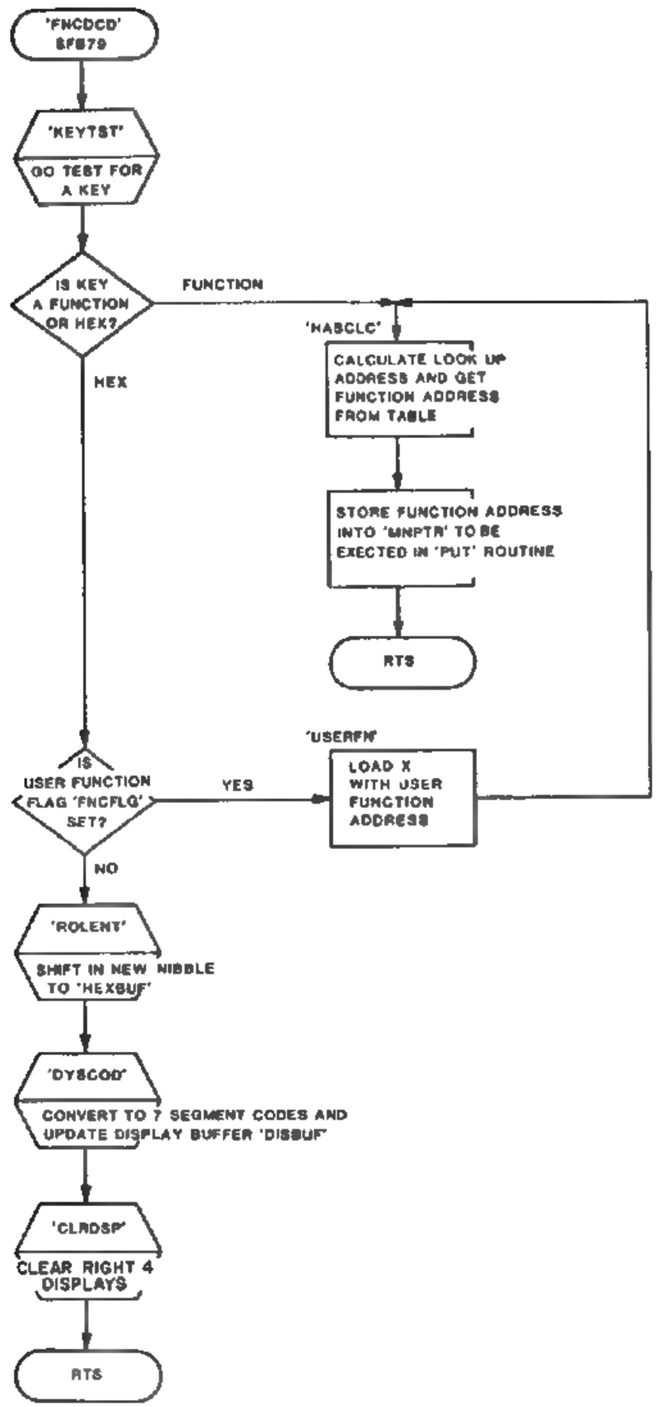


FIGURE 4.3.2. FUNCTION DECODE ROUTINE

4.4 Function Decode Routine (cont'd)

the HEX routine first tests FNCFLG which is set when the user function mode is active. If set, then the user function table address (UHASH) at $8102 is loaded into X. The HASCLC routine multiplies the key value by 2 and adds it to X to obtain the appropriate function address which is stored into MNPTR to be executed by PUT. If FNCFLG is not set, the routine assumes a valid hex key and calls ROLENT to shift the new value into HEXBUF. The DYSCOD routine is then called to convert the hex data from HEXBUF into seven-segment codes and stores them in DISBUF. The routine calls CLRDSP to clear the right four display digits and then returns to the PUT routine.

4.5 Interrupt Handling Routines

D3BUG handles all three types of 6802 interrupts: Maskable Interrupt Request (IRQ), Non-Maskable Interrupt (NMI), and Software Interrupt (SWI). In handling interrupts, the 6802 completes execution of its current instruction, saves the results on the stack, and then outputs the appropriate vector address. At that address it expects to find the beginning address of the selected interrupt service routine (for more detail, see the MC6802 data sheet in the appendix).

4.5.1 Maskable Interrupt Request

The IRQ interrupt is dedicated for the user. D3BUG handles the interrupt in the following manner. The user places his desired interrupt routine address in RAM at location $8106. When the interrupt occurs, the routine UIRQ located at $F95F is executed. This routine loads the contents of

4.5.1 Maskable Interrupt Request (cont'd)

location $81Ø6 and jumps to that address. The user must issue an RTI instruction at the end of this routine to return to the interrupted program.

4.5.2 Non-Maskable Interrupt

The NMI interrupt is used by the MEK6802D3 board to handle both keypad and trace interrupts. D3BUG will also handle user NMI's. A user NMI vector is provided at location $8104 called UNMIV. If the NMI service routine (NONMSK) determines that the interrupt is not a keypad or trace interrupt, it assumes it to be a user NMI and jumps to the program address stored at $8104. The user must issue an RTI instruction at the end of his routine to return to the interrupted program.

4.5.3 Software Interrupt

Software interrutps are used by D3BUG to implement breakpoints (up to a maximum of eight are allowed). D3BUG also permits the user to use the SWI through a vector location at $8108 called USWIV. The user must issue an RTI instruction at the end of his routine to return to the interrupted program.

4.6 User Callable Subroutines

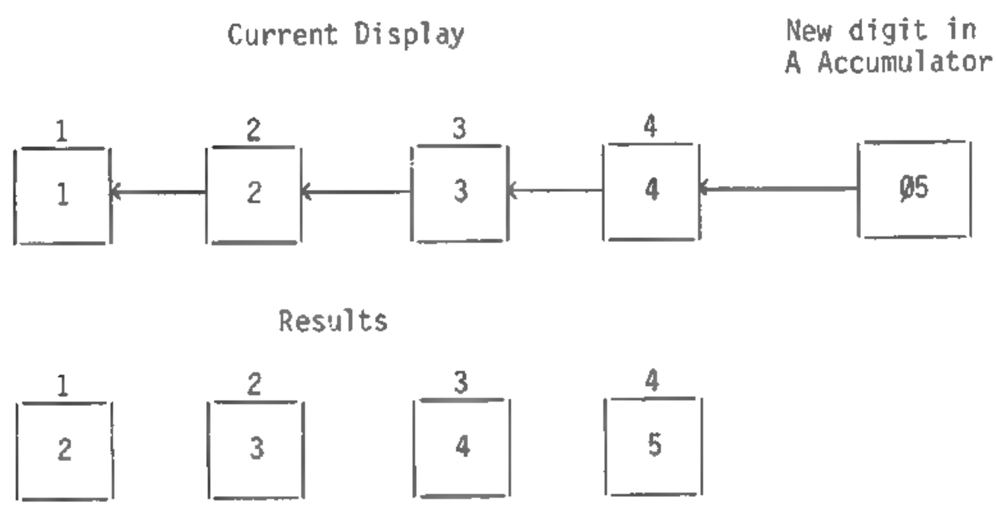
The majority of the routines written in D3BUG are callable subroutines. This allows them to be used by the expansion ROM or by the user. The following section describes some of the most commonly used subroutines. The D3BUG assembly listing located in the appendix should be referred to if additional explanation is required.

4.6.1 ROLENT

A callable subroutine located at $FB44 and used for special handling of hex numbers to be displayed in the seven-segment displays. This routine allows the insertion of a new hex digit to the hex display buffer shifting previous digits to the left. The routine can handle a four digit mode for displays 1-4 and a two digit mode for displays 5-6 by the use of FLG24 (located at $8135). If leading zeros are desired, ROLFLG (located at $8134) is cleared to zero. The digit to be stored must be stored in the A accumulator upon entry to the subroutine and the B accumulator must be saved if required later by the user. The following example is a user routine to enter a new number to the display.

<u>User Program Example</u>

```
86 Ø5      LDAA    #Ø5       Load 5 into accumulator
7F 8135    CLR     FLG24     Zero for 4 digit mode
7C 8134    INC     ROLFLG    Set to indicate not first entry
BD FB44    JSR     ROLENT    Store 5 in HEXBUF
BD FCØB    JSR     DYSCOD    Convert HEXBUF to display buffer
7E F9ØB    JMP     PUT       Return to D3BUG control and update
                             display
```



4.6.2 DYSCOD

A callable subroutine, located at $FC∅8, which converts HEXBUF (a hex display buffer) into seven-segment values and stores them in DISBUF (the display buffer). DISBUF is used to update the eight seven-segment displays. The user first stores the desired hex digits to be displayed in HEXBUF, which is a four byte buffer located at $8118 as shown below.

```
$8118     HEXBUF     RMB 4
```

1st Display	2nd Display	1st Byte
3rd Display	4th Display	2nd Byte
5th Display	6th Display	3rd Byte
7th Display	8th Display	4th Byte

Next the user performs a JSR to the DYSCOD subroutine. This routine will connect each hex value to its seven-segment value and store it in DISBUF. Upon completion, the program returns to the user. The next several times the PUT routine is executed the values stored in DISBUF are stored to the seven-segment displays.

4.6.3 ADDXA

A callable subroutine, located at $F94D, which will add the A accumulator to the X register and return to the caller with the new value in the X register with no change to the A or B accumulators.

4.6.4 EFPI

A callable subroutine, located at $F839, that will enable the keypad interrupts.

4.6.5 EXTST

A callable subroutine, located at \$F83F, that tests for the presence of the expansion ROM by checking address \$F000-1 for a \$55AA. If found the condition code will be set to a zero and returns to the caller. The caller can branch on this condition code.

4.6.6 CLRD

A callable subroutine, located at \$F846, which clears all 8 LED displays by storing an \$00 to each of the DISBUF locations in RAM. Individual displays can be cleared by loading the A accumulator with bits set to indicate which display to clear and entering at \$F848 (CLRDSP). The A accumulator and the X register are destroyed. The Display/Bit representation of accumulator A is shown below.

1	2	3	4	5	6	7	8
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7

LED Displays

4.6.7 DLY, DLY1, DLY25

A callable subroutine, located at \$F949, which provides a means to create calculatable program delays by loading the X register with the delay count, i.e., 109_{10} = 1 millisecond delay. DLY1, located at \$F946 will give a 1 millisecond delay. DLY25 located at \$F941 will give a 25 millisecond delay.

4.6.8 GET

A callable subroutine, located at \$F964, that is used by D3BUG to deter-

4.6.8 GET (cont'd)

mine which key was depressed as a result of an NMI keypad interrupt. This routine could be used in a users NMI interrupt routine to service a keypad interrupt.

4.6.9 ADDR

A callable subroutine, located at $FFD8, which subtracts two sixteen bits numbers with the answer stored in the A and B accumulator. To use the routine, first store the subtrahend at locations $813A (MSB) and $813B (LSB), then store the minuend at locations $813C (MSB) and $813D (LSB). Upon return from the JSR, the A accumulator holds the MSB and the B accumulator holds the LSB. The X register is destroyed.

4.6.10 DISNMI

A callable subroutine, located at $F8A4, which disables the keypad non-maskable interrupts. The A accumulator is destroyed.

4.6.11 ENNMI

A callable subroutine, located at $F8AF, which enables keypad NMI's.

4.6.12 GETIT

A callable subroutine, located at $FAC4, which interrogates the value in KEY. If the escape function is found, the program jumps to PROMPT; otherwise, the program checks whether the key is a hex or a function key. If a hex key is found, the MPU condition code N is set and the key value is placed in the A accumulator. If a function key is found, the Z condition code is cleared, bits 4-7 of the key value are masked off, and the result is placed in the A accumulator.

4.6.13 IRQ

A callable subroutine, located at $FFE4, that inputs and outputs to the cassette tape through an ACIA located at $8ØØA-B.

4.6.14 LOAD

A callable subroutine, located at $FED3, that loads from an ACIA at $8ØØA-B, a 1200 baud binary formatted tape.

4.6.15 S3ØØLD

A callable subroutine, located at $FEF5, that loads from an ACIA located at $8ØØA-B, a 300 baud K.C. Standard formatted tape.

4.6.16 S12ØØP

A callable subroutine, located at $FFB1, that punches a 1200 baud binary formatted tape to ACIA located at $8ØØA-B.

4.6.17 S3ØØP

A callable subroutine, located at $FF79, that punches a 300 baud K.C. Standard tape to an ACIA located at $8ØØA-B.

APPENDIX 1

D3BUG 1.0 ASSEMBLY LISTING

```
               NAM     D3BUG
               TTL     MOTOROLA INC.,'MAKING IT HAPPEN IN MEMORY S
               OPT     LLEN=120,CREF
* REV 1.0
* COPYRIGHT(C)1979 BY MOTOROLA INC.
* MEMORY SYSTEMS 'MAKING IT HAPPEN'
* AUSTIN,TEXAS
* MICROCOMPUTER CAPITAL OF THE WORLD! 3/30/79

*****************************************************
*
*   D3BUG 1.0 COMMAND SUMMARY
*
*
*  KEY    ADDRESS   DESCRIPTION
*
*  M        FAE2    COMMAND TO ALLOW MEMORY LOCATIONS
*                   TO BE DISPLAYED AND CHANGED.
*
*  EX       F85A    ESCAPE-ALLOWS D3BUG REENTRY.DOES
*                   NOT CLEAR BREAKPOINTS.
*
*  RD       F9E3    ALLOWS USER TO DISPLAY
*                   AND CHANGE USER REGISTERS.
*
*  GO       FC6A    GIVES CONTROL TO ADDRESS IN
*                   THE USER PROGRAM COUNTER.
*
*  FS       FC52    SETS THE SPECIAL FUNCTION FLAG.
*
*  FC       FC5F    CLEARS THE SPECIAL FUNCTION FLAG.
*
*  P/L      FE22    PUNCH/LOAD/VERIFY 1200/300 BAUD TAPE
*
*
*  T/B      FCD6    ALLOWS DISPLAYING,ADDING, AND
*                   DELETING UP TO 8 BREAKPOINTS.
*
```

```
**********************************************************
*
*  D3BUG CALLABLE SUBROUTINES
*
* ROUTINE ADDRESS DESCRIPTION
*
* ADDR    FFDE    SUBTRACTS TWO 16-BIT NUMBERS
*                 STORE IN 813A & 813C AND RETURNS
*                 RESULT IN 'A' & 'B'
*                 RESULT = (813C,813D)-(813A-813B).
*
* ADDXA   F953    ADDS THE A-ACCUMULATOR TO THE
*                 X-REGISTER.
*
* CLRD    F849    CLEARS DIGITS AS INDICATED BY THE
*                 BYTE IN THE A-ACCUMULATOR.
*
* DISNMI  F8A7    DISABLES KEYBOARD NMI'S.
*
* ENNMI   F8B2    ENABLES KEYBOARD NMI'S.
*
* DLY     F94F    DELAY LOOP WITH X REGISTER HOLDING
*                 LENGTH OF DELAY
*                 X=$6D=109=1MILLISECOND
*
* DLY1    F94C    DELAY 1 MILLISECOND
*
* DLY25   F947    DELAY 25 MILLISECONDS.
*
* DYSCOD  FC11    DECODES 4 HEX BYTES STORED FROM HEXBUF
*                 TO HEXBUF + 3 INTO 7-SEGMENT DISPLAY
*                 CODES WHICH ARE STORED FROM DISBUF TO
*                 DISBUF + 7 FOR DISPLAY.
*
* EFPI    F83C    ALLOWS A KEYBOARD INTERRUPT TO OCCUR
*                 WHEN ANY KEY IS PUSHED.
*
* EXTST   F842    LOOKS FOR THE PRESENCE OF $55AA
*                 AT $F000.  USED TO LOOK FOR AN EXPANSIO
*                 ROM.
*
* GETIT   FACA    READS THE KEYBOARD AND DETERMINES IF
*                 A FUNCTION KEY OR A HEX KEY WAS PUSHED.
*                 RETURNS TO D3BUG PROMPT MONITOR IF EX
*                 KEY IS PRESSED.
*
* IRQ     FFEA    INPUTS/OUTPUTS TO THE TAPE ACIA ($800B)
*
* LOAD    FED9    LOADS FROM A 1200 BAUD FORMATTED TAPE.
*
* S300LD  FEFB    LOADS FROM A 300 BAUD FORMATTED TAPE.
*
* S1200P  FFB7    PUNCH A 1200 BAUD FORMATTED TAPE.
*
* S300P   FF7F    PUNCH A 300 BAUD FORMATTED TAPE.
*
* ROLENT  FB4A    PACKS HEX BYTES INTO ONE OR TWO BYTES.
```

```
          817F   A STKTOP EQU    $817F     TOP OF STACK
          8088   A DISREG EQU    $8088     PORT A KEYBOARD/DISPLAY PIA
          8089   A DISCTR EQU    $8089     PORT A CONTROL REG
          808A   A SCNREG EQU    $808A     PORT B
          808B   A SCNCTR EQU    $808B     PORT B CONTROL REG
          8085   A TCNTRL EQU    $8085     6846 CONTROL REG.
          8086   A TMRMSB EQU    $8086     6846 TIMER REG.
          800A   A ACIA0  EQU    $800A     CASSETTE CONTROL REG.
          800B   A ACIA1  EQU    $800B     CASSETTE DATA REG.
          F000   A EXROM  EQU    $F000     EXPANSION ROM STARTING ADDR.
          F002   A EXSTRT EQU    EXROM+2   RESTART JUMP FOR EXPANSION ROM
          F005   A EXREEN EQU    EXSTRT+3  REENTRY JUMP
          F008   A EXGET  EQU    EXREEN+3  GET CHARACTER FROM ASCII KEYBOARD
          F00B   A EXPUT  EQU    EXGET+3   PUT CHARACTER ON R2 DISPLAY
          F00E   A EXKEY  EQU    EXPUT+3   TEST FOR A ASCII KEY PUSHED


                 ***********************************************************
                 *
                 * D3BUG 1.0 INITIALIZATION SEQUENCE
                 * BEGINS HERE
                 *
                 ***********************************************************

A F800                    ORG    $F800
A F800 8E 817F     RESTRT LDS    #STKTOP   INITIALIZE STACK POINTER
A F803 0F                 SEI              PUT OUT DO NOT DISTURB SIGN
A F804 CE 00FF   A        LDX    #$00FF    DEFAULT VALUE FOR USER STACK
A F807 FF 812B   A        STX    USP
A F80A CE 000E   A        LDX    #$E
A F80D FF 8086   A        STX    TMRMSB    LOAD TIMER WITH LENGTH OF TRACE
A F810 7F 812D   A        CLR    INIT1     FRESH START FLAG
A F813 CE 8143   A        LDX    #BOF
A F816 FF 8140   A        STX    TOFT      INITIALIZE TOP OF D3BREAK TABLE
A F819 4F                 CLRA
A F81A B7 8081   A        STAA   $8081     INITIALIZE PIA
A F81D 43                 COMA
A F81E B7 8082   A        STAA   $8082
A F821 B7 808A   A        STAA   SCNREG
A F824 44                 LSRA
A F825 B7 8088   A        STAA   DISREG
A F828 7F 8083   A        CLR    $8083     INITIALIZE PAGE SELECT TO 0
A F82B 8D 77 F8A4         BSR    DISNMI
                 *
                 * LOOK FOR EXPANSION ROM
                 *
A F82D 8D 10 F83F         BSR    EXTST     GO CHECK FOR EXPANSION ROM
A F82F 26 26 F857         BNE    PROMPT    IF NONE, GO ON WITH D3BUG
A F831 8D 06 F839         BSR    EFPI      ELSE,ENABLE KEYPAD INTERRUPTS
A F833 B7 8130   A        STAA   EXFLAG    SET EXPANSION ROM PRESENCE FLAG
A F836 7E F002   A        JMP    EXSTRT    AND JUMP TO EXPANSION ROM
```

```
                        *
                        * SUBROUTINE TO ENABLE KEY PAD INTERRUPTS
                        *
A F839 86 3F     A EFPI    LDAA   #$3F
A F83B B7 808A   A         STAA   SCNREG
A F83E 39                  RTS                **RETURN**


                        *
                        * SUBROUTINE TO CHECK FOR EXPANSION ROM
                        *
A F83F FE F000   A EXTST   LDX    EXROM
A F842 8C 55AA   A         CPX    #$55AA
A F845 39                  RTS                **RETURN**


                        *
                        * SUBROUTINE TO CLEAR DISPLAY
                        *
A F846 86 FF     A CLRD    LDAA   #$FF      CLEAR THEM ALL-
A F848 CE 811C   A CLRDSP  LDX    #DISBUF   OR CLEAR SELECTED DIGITS
A F84B 44          LOP     LSRA             WHICH DISPLAYS TO ERASE?
A F84C 24 02 F850          BCC    SKIP11    THOSE WHICH ARE SET IN 'A'
A F84E 6F 00     A         CLR    0,X
A F850 08          SKIP11  INX
A F851 8C 8124   A         CPX    #DISBUF+8 LAST DISPLAY?
A F854 26 F5 F84B          BNE    LOP       NO, CONTINUE CLEARING.
A F856 39                  RTS              **RETURN**


                        ***************************************************************
                        *
                        * PROMPT ROUTINE-INITIALIZE DISPLAYS
                        *
                        ***************************************************************
A F857 8D ED F846 PROMPT   BSR    CLRD
A F859 B6 808A   A         LDAA   SCNREG    CLEAR KEY NMI
A F85C 86 41     A         LDAA   #$41      7-SEG CODE FOR PROMPT (=)
A F85E B7 811C   A         STAA   DISBUF
A F861 86 80     A PROMPG  LDAA   #$80
A F863 B7 812E   A         STAA   ROWCOL    INDICATE NO KEY PENDING
A F866 CE 812F   A         LDX    #NFLAG
A F869 6F 00     A LOOP1   CLR    0,X
A F86B 08                  INX
A F86C 8C 8138   A         CPX    #INIT2+1
A F86F 26 F8 F869          BNE    LOOP1
                        *
                        * INITIALIZE FLAGS ETC.
                        *
A F871 8E 817F   A PROMP1  LDS    #STKTOP   INITIALIZE STACK POINTER
A F874 CE FB79   A         LDX    #FNCDCD   GET FUNCTION DECODE ADDRESS
A F877 FF 8100   A         STX    MNPTR     INITIALIZE 'MAIN' POINTER
A F87A 8D 33 F8AF          BSR    ENNMI     ENABLE KEYPAD NMI'S
A F87C 7E F90B   A         JMP    PUT       BEGIN EXECUTION
```

```
00202                    *****************************************************
00203                    *
00204                    *  NMI INTERRUPT ENTRY POINT
00205                    *
00206                    *****************************************************
00207A F87F 86 B3     A NONMSK LDAA   #$B3      PRESET TIMER
00208A F881 B7 8085   A        STAA   TCNTRL
00209A F884 4F                 CLRA
00210A F885 B7 8088   A        STAA   DISREG    TURN OFF LEDS
00211A F888 B6 8089   A        LDAA   DISCTR    CHECK TRACE INTERRUPT FLAG
00212A F88B 2B 3F F8CC         BMI    CONTIN    IF TRACE, GO TO TRACE ROUTINE
00213A F88D 8D 15 F8A4         BSR    DISNMI
00214A F88F B6 808B   A        LDAA   SCNCTR    CHECK KEY INTERRUPT
00215A F892 2B 05 F899         BMI    KEYNMI    IF KEY NMI,GO TO KEY ROUTINE
00216A F894 FE 8104   A        LDX    UNMIV     ELSE,MUST BE USER NMI-GET VECTOR
00217A F897 6E 00     A        JMP    0,X       AND GO TO USER ROUTINE
00218A F899 7D 8130   A KEYNMI TST    EXFLAG
00219A F89C 26 B9 F857         BNE    PROMPT
00220A F89E BD F964   A        JSR    GET       GO GET & DECODE THE KEY VALUE
00221A F8A1 8D 0C F8AF         BSR    ENNMI     ENABLE INTERRUPTS
00222A F8A3 3B                 RTI              **INTERRUPT SEQUENCE DONE**


00224                    *
00225                    * SUBROUTINE TO DISABLE KEYPAD NMI'S
00226                    *
00227A F8A4 86 3E     A DISNMI LDAA   #$3E      CB1 RISING EDGE TRIGGER
00228A F8A6 B7 808B   A NMIX   STAA   SCNCTR    PORT B CONTROL REG
00229A F8A9 88 02     A        EORA   #2        CA1 FALLING EDGE TRIGGER
00230A F8AB B7 8089   A        STAA   DISCTR    PORT A CONTROL REG
00231A F8AE 39                 RTS              **RETURN**


00233                    *
00234                    * SUBROUTINE TO ENABLE KEYPAD NMI'S
00235                    *
00236A F8AF 86 3F     A ENNMI  LDAA   #$3F
00237A F8B1 20 F3 F8A6         BRA    NMIX      EXIT WILL OCCUR THERE
```

```
                          *************************************************************
                          *
                          * SOFTWARE INTERRUPT ENTRY POINT
                          *
                          *************************************************************
A F8B3 30                 SWINT  TSX
A F8B4 A6 06      A              LDAA   6,X
A F8B6 80 01      A              SUBA   #1
A F8B8 B7 8119    A              STAA   HEXBUF+1
A F8BB A6 05      A              LDAA   5,X
A F8BD 82 00      A              SBCA   #0
A F8BF B7 8118    A              STAA   HEXBUF    SAVE MEMORY LOCATION FOR FNDBRK
A F8C2 BD FD88    A              JSR    FNDBRK
A F8C5 27 05 F8CC                BEQ    CONTIN    IF NORMAL BRKPT,GO TO BRKPT ROUTIN
A F8C7 FE 8108    A              LDX    USWIV     ELSE,GET USER SWI VECTOR
A F8CA 6E 00      A              JMP    0,X       & GIVE CONTROL TO HIS ROUTINE
A F8CC CE 8124    A CONTIN LDX    #UCC      POINT AT USER REG STORAGE AREA
A F8CF 32                 LOOP10 PULA             GET STACKED USER REG VALUE
A F8D0 A7 00      A              STAA   0,X       PUT IN CORRECT LOCATION
A F8D2 08                        INX
A F8D3 8C 812B    A              CPX    #UCC+7    SEE IF DONE
A F8D6 26 F7 F8CF                BNE    LOOP10    IF NOT; CONTINUE
A F8D8 BF 812B    A              STS    USP       SAVE USER'S STACK POINTER
A F8DB 8E 817F    A              LDS    #STKTOP   PUT STACK BACK TO D3BUG STACK
A F8DE 7F 8132    A              CLR    INIT      FIRST PASS FLAG USED BY REGDIS
A F8E1 B6 812F    A              LDAA   NFLAG     IN TRACE MODE?
A F8E4 47                        ASRA
A F8E5 25 14 F8FB                BCS    TRCE      YES, A TRACE HAS OCCURRED
A F8E7 BD FDFD    A BKP          JSR    RMBKPT    NO,MUST BE A BRKPT-REMOVE IT
A F8EA FE 8129    A              LDX    UPC
A F8ED 09                        DEX
A F8EE A6 00      A              LDAA   0,X
A F8F0 81 3F      A              CMPA   #$3F
A F8F2 26 03 F8F7                BNE    ARND1
A F8F4 FE 8108    A              LDX    USWIV
A F8F7 FF 8129    A ARND1        STX    UPC
A F8FA 4F                        CLRA
A F8FB CE F9DD    A TRCE         LDX    #REGDIS   REGISTER DISPLAY IS DEFAULT
A F8FE 48                        ASLA             RETURN SIGN BIT-CLEAR TRACE FLAG
A F8FF 2A 04 F905                BPL    MAIN
A F901 4F                 BKPCNT CLRA             CLEAR ALL FLAGS
A F902 CE FC64    A              LDX    #GO       CONTINUE EXECUTING PROGRAM
A F905 B7 812F    A MAIN         STAA   NFLAG     SET FLAGS
A F908 FF 8100    A              STX    MNPTR     PROGRAM DESIRED TO RUN
```

```
                     ****************************************************
                     *
                     * PUT ROUTINE USED TO DISPLAY NUMBERS
                     *  AND GO TO DESIRED COMMAND ROUTINE
                     *
                     ****************************************************
A F90B BD F83F   A PUT    JSR    EXTST    EXPANSION ROM PRESENT?
A F90E 26 03 F913         BNE    PUT2     NO, CONTINUE NORMAL OPERATION
A F910 BD F00E   A        JSR    EXKEY    CHECK FOR AN ASCII KEY
A F913 7F 8130   A PUT2   CLR'   EXFLAG   CLEAR THE RETURN FROM'GO'FLAG
A F916 86 80     A        LDAA   #$80
A F918 B7 8115   A        STAA   DIGEN    INITIALIZE DIGIT ENABLE WORD
A F91B B6 8115   A LP1    LDAA   DIGEN
A F91E B7 808A   A        STAA   SCNREG   STORE IT TO PIA
A F921 CE 811B   A        LDX    #DISBUF-1 POINT AT DISPLAY
A F924 08          LP2    INX
A F925 48                 ASLA
A F926 24 FC F924         BCC    LP2      POINT X-REG AT ACTIVE DISPLAY
A F928 A6 00     A        LDAA   0,X      GET VALUE TO BE DISPLAYED
A F92A B7 8088   A        STAA   DISREG   STORE TO DISPLAY
A F92D 8D 17 F946         BSR    DLY1     DELAY 1 MS.
A F92F 7F 8088   A        CLR    DISREG   TURN OFF DISPLAY
A F932 BD F839   A        JSR    EFPI
A F935 FE 8100   A        LDX    MNPTR    GET ADDRESS IN "MAIN"
A F938 AD 00     A        JSR    0,X      EXECUTE COMMAND SUBROUTINE
A F93A 74 8115   A        LSR    DIGEN    SHIFT ACTIVE DISPLAY BIT
A F93D 26 DC F91B         BNE    LP1      CONTINUE TILL BIT FALLS IN CARRY
A F93F 20 CA F90B         BRA    PUT      REINITIALIZE DIGEN-CONTINUE
                     *
                     * DELAY SUBROUTINE
                     *
A F941 CE 0AEA   A DLY25  LDX    #2794    25 MS ENTRY POINT
A F944 20 03 F949         BRA    DLY
A F946 CE 006D   A DLY1   LDX    #109     1 MS ENTRY POINT
A F949 09          DLY    DEX
A F94A 26 FD F949         BNE    DLY      LOOP TILL X EQUAL ZERO
A F94C 39                 RTS             ** RETURN **
                     *
                     * SUBROUTINE WHICH DOES X=X+A
                     *
A F94D FF 8113   A ADDXA  STX    XCALC    PUT X-REG IN RAM
A F950 BB 8114   A        ADDA   XCALC+1  ADD A-REG TO LOW BYTE OF X
A F953 B7 8114   A        STAA   XCALC+1  UPDATE LOW BYTE
A F956 24 03 F95B         BCC    ARND     IF NO CARRY YOU'RE DONE
A F958 7C 8113   A        INC    XCALC    COMPENSATE FOR OVERFLOW
A F95B FE 8113   A ARND   LDX    XCALC    LOAD X-REG WITH X+A
A F95E 39                 RTS             ** RETURN **
                     *
                     * USER IRQ ENTRY ROUTINE
                     *
A F95F FE 8106   A UIRQ   LDX    UIRQV
A F962 6E 00     A        JMP    0,X      GIVE CONTROL TO USER IRQ PROGRAM
```

```
00337                          ************************************************************
00338                          *
00339                          * GET ROUTINE TO DETERMINE WHICH KEY PRESSED
00340                          *
00341                          ************************************************************
00342A F964 7D 8130   A GET    TST    EXFLAG    EXPANSION ROM PRESENT?
00343A F967 2F 03 F96C         BLE    GET1      NO, CONTINUE NORMAL OPERATION
00344A F969 7E F008   A        JMP    EXGET     ELSE,GO TO EXPANSION ROM
00345A F96C CE 8088   A GET1   LDX    #DISREG   POINT AT PIA
00346A F96F 86 3F     A        LDAA   #$3F
00347A F971 A7 02     A COLUMN STAA   2,X       TURN ON ALL ROWS
00348A F973 6D 00     A        TST    0,X       CHECK FOR KEY CLOSURE
00349A F975 2A 08 F97F         BPL    COLFND    IF KEY FOUND,GET COLUMN #
00350A F977 A6 02     A        LDAA   2,X
00351A F979 8B 40     A        ADDA   #$40
00352A F97B 24 F4 F971         BCC    COLUMN    KEEP TRYING TILL SURE NO KEY CLOSE
00353A F97D 20 E5 F964         BRA    GET
00354A F97F 84 C0     A COLFND ANDA   #$C0      KEEP ONLY COLUMN #
00355A F981 B7 810A   A        STAA   GTEMP     SAVE FOR REFERENCE
00356A F984 86 01     A        LDAA   #1        PREPARE TO SCAN ROWS
00357A F986 5F                 CLRB             START ROW COUNT
00358A F987 BA 810A   A ROW    ORAA   GTEMP     COMBINE ROW & COLUMN
00359A F98A A7 02     A        STAA   2,X       STORE TO PIA
00360A F98C 6D 00     A        TST    0,X       CHECK FOR CLOSURE
00361A F98E 2A 0A F99A         BPL    ROWFND    BRANCH IF ROW FOUND
00362A F990 84 3F     A        ANDA   #$3F      MASK OFF COLUMN
00363A F992 81 20     A        CMPA   #$20      MAKE SURE ROW  NOT EXCEEDED
00364A F994 27 CE F964         BEQ    GET       KEY BOUNCED; KEEP LOOKING
00365A F996 48                 ASLA             PREPARE TO LOOK AT NEXT ROW
00366A F997 5C                 INCB             INCREMENT ROW COUNT
00367A F998 20 ED F987         BRA    ROW       LOOP TILL ROW FOUND
00368A F99A 44          ROWFND LSRA             RIGHT JUSTIFY COLUMN
00369A F99B 44                 LSRA
00370A F99C 84 F0     A        ANDA   #$F0      MASK OFF LEFT NIBBLE FOR ROW
00371A F99E 1B                 ABA              CONCATENATE COLUMN/ROW
00372A F99F B7 812E   A        STAA   ROWCOL    SAVE RESULT
00373A F9A2 6D 00     A CLOP   TST    0,X       SEE IF KEY STILL CLOSED
00374A F9A4 2A FC F9A2         BPL    CLOP      STAY IN LOOP TILL KEY GOES AWAY
00375A F9A6 16          KEYCOD TAB              COPY ROW/COLUMN INFO
00376A F9A7 C4 07     A        ANDB   #$07      MASK OFF ALL BUT COLUMN
00377A F9A9 84 30     A        ANDA   #$30      MASK OFF ALL BUT ROW
00378A F9AB 44                 LSRA             MOVE ROW OVER AGAINST COLUMN
00379A F9AC 1B                 ABA              COMBINE TO FORM TABLE OFFSET
00380A F9AD CE F9BF   A        LDX    #KEYTBL   POINT AT TOP OF TABLE
00381A F9B0 BD F94D   A        JSR    ADDXA     FORM ADDRESS OF LOOK-UP DATA
00382A F9B3 A6 00     A        LDAA   0,X       GET CODE FOR KEY VALUE
00383A F9B5 B7 8117   A        STAA   KEY       STORE VALUE FOR USE BY OTHERS
00384A F9B8 BD F941   A        JSR    DLY25     DELAY 25 MS TO DEBOUNCE KEY
00385A F9BB F6 808A   A        LDAB   SCNREG    RESET NMI ON THE PIA
00386A F9BE 39                 RTS              ** RETURN **
00387                          * MSB OF ROWCOL IS SET IF NO KEY
```

```
00389                    *
00390                    * KEY DECODE LOOK-UP TABLE
00391                    *
00392A F9BF    00     A KEYTBL FCB    $00       '0'
00393A F9C0    01     A        FCB    $01       '1'
00394A F9C1    04     A        FCB    $04       '4'
00395A F9C2    07     A        FCB    $07       '7'
00396A F9C3    14     A        FCB    $14       'FS' FUNCTION SET
00397A F9C4    10     A        FCB    $10       'MD' MEMORY DISPLAY
00398A F9C5    00     A        FCB    $00       NOT USED
00399A F9C6    00     A        FCB    $00       NOT USED
00400A F9C7    0F     A        FCB    $0F       'F'
00401A F9C8    02     A        FCB    $02       '2'
00402A F9C9    05     A        FCB    $05       '5'
00403A F9CA    08     A        FCB    $08       '8'
00404A F9CB    15     A        FCB    $15       'FC' FUNCTION CLEAR
00405A F9CC    11     A        FCB    $11       'EX' ESCAPE
00406A F9CD    00     A        FCB    $00       NOT USED
00407A F9CE    00     A        FCB    $00       NOT USED
00408A F9CF    0E     A        FCB    $0E       'E'
00409A F9D0    03     A        FCB    $03       '3'
00410A F9D1    06     A        FCB    $06       '6'
00411A F9D2    09     A        FCB    $09       '9'
00412A F9D3    16     A        FCB    $16       'P/L' PUNCH/LOAD
00413A F9D4    12     A        FCB    $12       'RD' REGISTER DISPLAY
00414A F9D5    00     A        FCB    $00       NOT USED
00415A F9D6    00     A        FCB    $00       NOT USED
00416A F9D7    0D     A        FCB    $0D       'D'
00417A F9D8    0C     A        FCB    $0C       'C'
00418A F9D9    0B     A        FCB    $0B       'B'
00419A F9DA    0A     A        FCB    $0A       'A'
00420A F9DB    17     A        FCB    $17       'T/B' TRACE/BREAKPOINT
00421A F9DC    13     A        FCB    $13       'GO' GO TO USER PROG
```

```
                         ***********************************************************
                         *
                         * REGISTER DISPLAY ROUTINE
                         *
                         ***********************************************************
A F9DD 7D 8132  A REGDIS TST    INIT      FIRST PASS ?
A F9E0 26 08 F9EA        BNE    REGLOP    NO; BYPASS INITIALIZATION
A F9E2 7C 8132  A        INC    INIT
A F9E5 7F 8116  A        CLR    REGNUM    START WITH 'PC' DISPLAY
A F9E8 20 78 FA62        BRA    REGAD1    GO TO FIRST PASS ENTRY POINT
A F9EA B6 812E  A REGLOP LDAA   ROWCOL    SEE IF ANY KEY DOWN
A F9ED 81 15    A        CMPA   #$15      WAS IT 'E' KEY ?
A F9EF 26 05 F9F6        BNE    ARN1M     NO, FIND WHICH KEY IT IS
A F9F1 8D 58 FA4B        BSR    REGADV    YES,STOP ANY PENDING OPERATION
A F9F3 7E F857  A        JMP    PROMPT    ESCAPE TO PROMPT ROUTINE
A F9F6 BD FB22  A ARN1M  JSR    KEYTST    EXITS IF 'E' OR NO KEY
A F9F9 2B 18 FA13        BMI    NUMKEY    N CODE SET IF IT WAS A HEX KEY
A F9FB 81 02    A        CMPA   #2        IF NOT HEX,CHECK FOR 'RD' KEY
A F9FD 27 4C FA4B        BEQ    REGADV    BRANCH TO ADVANCE REGISTER
A F9FF 81 03    A        CMPA   #3        CHECK FOR GO KEY
A FA01 26 07 FA0A        BNE    TRAC      IF NOT'GO",CHECK TRACE COMMAND
A FA03 86 80    A        LDAA   #$80
A FA05 B7 812F  A        STAA   NFLAG     MAKE NFLAG NEG
A FA08 20 04 FA0E        BRA    TRAC1     THIS MEANS CONTINUE FROM A BRKPT
A FA0A 81 07    A TRAC   CMPA   #7        WAS IT"T"KEY?
A FA0C 26 08 FA16        BNE    DUNPAS    NO,EXIT THIS PASS
A FA0E 8D 3B FA4B TRAC1  BSR    REGADV    YES,COMPLETE PRESENT OPERATION
A FA10 7E FCD5  A        JMP    TRACE     & GO TO TRACE PROGRAM
A FA13 BD FB44  A NUMKEY JSR    ROLENT    KEY IS HEX,SO SHIFT INTO HEXBUF
A FA16 7D 8116  A DUNPAS TST    REGNUM    KEY PROCESSED SO UPDATE & EXIT
A FA19 26 08 FA23        BNE    PAST1
A FA1B FE 8118  A        LDX    HEXBUF    'PC' MODE SO:
A FA1E A6 00    A        LDAA   0,X       GET OP-CODE,
A FA20 B7 811A  A        STAA   HEXBUF+2  STORE TO HEXBUF
A FA23 BD FC0B  A PAST1  JSR    DYSCOD    CONVERT HEXBUF TO 7-SEGMENT
A FA26 B6 8116  A        LDAA   REGNUM
A FA29 26 07 FA32        BNE    PAST2     IF NOT"PC"CHECK FOR ANOTHER KEY
A FA2B CE 7339  A        LDX    #$7339    7-SEGMENT CODE FOR "PC"
A FA2E FF 8122  A        STX    DISBUF+6  PUT IN DIGITS 7 & 8
A FA31 39                RTS              **RETURN**
A FA32 81 05    A PAST2  CMPA   #5        CHECK FOR "SP"
A FA34 26 06 FA3C        BNE    BLANK     IF NOT,GO TO BLANKING ROUTINE
A FA36 CE 6D73  A        LDX    #$6D73    7-SEGMENT CODE FOR "SP"
A FA39 FF 8122  A        STX    DISBUF+6  PUT IN DIGITS 7 & 8
A FA3C 84 03    A BLANK  ANDA   #$03      A-REG CONTAINS REGNUM
A FA3E 4A                DECA
A FA3F 26 04 FA45        BNE    CLR14     IF A-REG NOT 0,BLANK FIRST 4
A FA41 86 30    A        LDAA   #%00110000 CLEAR DIGITS 5 & 6
A FA43 20 02 FA47        BRA    CLR1      OK YOU'RE DONE SO RETURN
A FA45 86 0F    A CLR14  LDAA   #%00001111 BLANK FIRST FOUR DISPLAYS
A FA47 BD F848  A CLR1   JSR    CLRDSP    GO CLEAR DISPLAY
A FA4A 39                RTS              **RETURN**
```

```
00477A FA4B FE 8118  A REGADV LDX    HEXBUF   DONE HERE TO SAVE CODE
00478A FA4E B6 811A  A        LDAA   HEXBUF+2 A-REG OR X-REG WILL BE NEEDED
00479A FA51 F6 8116  A        LDAB   REGNUM   GET CURRENT REGISTER NUMBER
00480A FA54 5C                INCB            ADV TO NEXT REGISTER
00481A FA55 F7 8116  A        STAB   REGNUM   UPDATE REGISTER NUMBER
00482A FA58 C1 06    A        CMPB   #6       CHECK FOR OVERRUN
00483A FA5A 26 0E FA6A        BNE    ADV1     NOT WRAP AROUND (SO NOT 'PC')
00484A FA5C 7F 8116  A        CLR    REGNUM   LAST REG WAS'SP'SO RESET TO'PC'
00485A FA5F FF 812B  A        STX    USP      UPDATED SP WAS IN X-REG; SAVE IT
00486A FA62 FE 8129  A REGAD1 LDX    UPC      GET STORED USER PROG COUNTER
00487A FA65 FF 8118  A        STX    HEXBUF   PUT IN WORKING/DISPLAY BUFFER
00488A FA68 20 55 FABF        BRA    OUT2     EXIT BY WAY OF 2-BYTE OPTION
00489A FA6A 5A         ADV1   DECB            REGNUM  WILL BE DEC'D TO 0
00490A FA6B 26 10 FA7D        BNE    ADV2     WASN'T 'ID' SO KEEP LOOKING
00491A FA6D FF 8129  A        STX    UPC      X-REG HAD UPDATED USER PC
00492A FA70 FE 8127  A        LDX    UX       GET STORED USER X-REG VALUE
00493A FA73 FF 8118  A        STX    HEXBUF   PUT IN DISPLAY/CHANGE BUFFER
00494A FA76 86 1D    A        LDAA   #$1D     7-SEG CODE FOR 'ID'
00495A FA78 B7 811B  A        STAA   HEXBUF+3 DYSCOD CONVERTS TO 7-SEGMENT
00496A FA7B 20 42 FABF        BRA    OUT2     EXIT WITH 2-BYTE OPTION
00497A FA7D 5A         ADV2   DECB
00498A FA7E 26 16 FA96        BNE    ADV3     IF NOT 'AA' MODE GO ON
00499A FA80 FF 8127  A        STX    UX       UPDATE USER X-REG
00500A FA83 B6 8126  A        LDAA   UA       GET USER'S A-REG
00501A FA86 B7 811A  A        STAA   HEXBUF+2 PUT IN LOCATIONS 5&6
00502A FA89 86 AA    A        LDAA   #$AA     CODE FOR 7TH/8TH DIGITS
00503A FA8B B7 811B  A SOUT1  STAA   HEXBUF+3
00504A FA8E B7 8135  A OUT1   STAA   FLG24    SET TO 2 NIBBLE MODE
00505A FA91 7F 8134  A DUNADV CLR    ROLFLG   SET FIRST PASS FLAG FOR ROLENT
00506A FA94 20 80 FA16        BRA    DUNPAS   GO UPDATE THE DISPLAY
00507A FA96 5A         ADV3   DECB
00508A FA97 26 0D FAA6        BNE    ADV4     IF NOT 'BB' GO AROUND
00509A FA99 B7 8126  A        STAA   UA       UPDATE USER'S A-REG
00510A FA9C B6 8125  A        LDAA   UB       GET HIS B-REG
00511A FA9F B7 811A  A        STAA   HEXBUF+2 PREPARE TO DISPLAY/CHANGE IT
00512A FAA2 86 AB    A        LDAA   #$AB     CODE FOR DIGITS 7 & 8
00513A FAA4 20 E5 FA8B        BRA    SOUT1    STORE CODE-EXIT VIA 1-BYTE OPT
00514A FAA6 5A         ADV4   DECB
00515A FAA7 26 0D FAB6        BNE    ADV5
00516A FAA9 B7 8125  A        STAA   UB       UPDATE
00517A FAAC B6 8124  A        LDAA   UCC      GET USER'S COND CODE REG
00518A FAAF B7 811A  A        STAA   HEXBUF+2
00519A FAB2 86 CC    A        LDAA   #$CC
00520A FAB4 20 D5 FA8B        BRA    SOUT1
00521A FAB6 B7 8124  A ADV5   STAA   UCC      BY DEFAULT MUST BE SP MODE
00522A FAB9 FE 812B  A        LDX    USP
00523A FABC FF 8118  A        STX    HEXBUF   PREPARE SP FOR DISP/CHNG
00524A FABF 7F 8135  A OUT2   CLR    FLG24    SET TO 4 NIBBLE MODE
00525A FAC2 20 CD FA91        BRA    DUNADV
```

```
                         *
                         * TEST FOR HEX/FUNCTION KEY AND PUT KEY VALUE IN A
                         *
A FAC4 78 812E  A GETIT  ASL     ROWCOL
A FAC7 0D                SEC
A FAC8 76 812E  A        ROR     ROWCOL   SET BIT 7 TO ACKNOWLEDGE KEY
A FACB B6 8117  A        LDAA    KEY      NOW GET THE DECODED KEY
A FACE 81 10    A        CMPA    #$10     CHECK FOR HEX #/FUNCTION KEY
A FAD0 2B 09 FADB        BMI     RET      IF N CODE SET KEY WAS HEX
A FAD2 81 11    A        CMPA    #$11     CHECK FOR ESCAPE KEY
A FAD4 26 03 FAD9        BNE     ARN1     IF NOT GO AROUND
A FAD6 7E F857  A        JMP     PROMPT   EXIT TO PROMPT SEQUENCE
A FAD9 84 0F    A ARN1   ANDA    #$0F     CONDENSE FUNCTION # TO 0 - 7
A FADB 39         RET    RTS              ** RETURN **
                         * N CONDITION CODE INDICATES KEY TYPE


                         *************************************************
                         *
                         * MEMORY CHANGE ROUTINE
                         *
                         *************************************************
A FADC 7D 8137  A MEMCH  TST     INIT2
A FADF 26 51 FB32        [redacted] OFFSET
A FAE1 FE 8118  A        LDX     HEXBUF   POINT AT MEM LOC SPECIFIED
A FAE4 FF 8111  A        STX     XTMP3
A FAE7 7D 8132  A        TST     INIT     SEE IF FIRST PASS
A FAEA 26 08 FAF4        BNE     BEGIN    IF NOT BYPASS INITAILIZATION
A FAEC 7C 8132  A        INC     INIT     INDICATE NOT FIRST PASS
A FAEF 7C 8135  A        INC     FLG24    SET TO 2-NIBBLE MODE
A FAF2 20 1B FB0F        BRA     FUNDIS   SET UP DISPLAY
A FAF4 8D 2C FB22 BEGIN  BSR     KEYTST   GO SEE IF KEY PENDING
A FAF6 2A 09 FB01        BPL     FUNKEY   IF FUNCTION KEY SEE WHICH ONE
A FAF8 8D 4A FB44        BSR     ROLENT   IF # KEY UPDATE HEXBUF
A FAFA B6 811A  A        LDAA    HEXBUF+2 GET VALUE ENTERED
A FAFD A7 00    A        STAA    0,X      STORE (OR AT LEAST TRY TO)
A FAFF 20 19 FB1A        BRA     DISP     LEAVE VIA DISPLAY
A FB01 26 03 FB06 FUNKEY BNE     OFF
A FB03 09                DEX              M KEY MEANS BACK UP 1 LOCATION.
A FB04 20 09 FB0F        BRA     FUNDIS
A FB06 81 04    A OFF    CMPA    #4
A FB08 27 28 FB32        BEQ     OFFSET
A FB0A 81 03    A GOKEY  CMPA    #3       SEE IF G KEY
A FB0C 26 0C FB1A        BNE     DISP     IF NOT EXIT
A FB0E 08                INX              G KEY MEANS ADVANCE TO NEXT LOC
A FB0F FF 8118  A FUNDIS STX     HEXBUF   GET NEW ADDR INTO HEXBUF
A FB12 A6 00    A        LDAA    0,X      GET DATA FROM NEW LOC
A FB14 B7 811A  A        STAA    HEXBUF+2 INITIAL VAL TO DISPLAY
A FB17 7F 8134  A        CLR     ROLFLG   SET ROLENT FIRST PASS FLAG
A FB1A A6 00    A DISP   LDAA    0,X      GET ACTUAL VALUE AT LOC
A FB1C B7 811B  A        STAA    HEXBUF+3 STORE TO DIGITS 7 & 8
A FB1F 7E FC08  A        JMP     DYS
```

```
00579                        *
00580                        *  GET KEY VALUE AND RETURN
00581                        *
00582A FB22 7D 8130   A KEYTST TST    EXFLAG   EXPANSION ROM PRESENT?
00583A FB25 2F 03 FB2A         BLE    SKIP1    NO, CONTINUE NORMAL OPERATION.
00584A FB27 7E F008   A        JMP    EXGET    EXGET-RETURN TO CALLING PROGRAM.
00585A FB2A 7D 812E   A SKIP1  TST    ROWCOL   CHECK FOR NEW KEY
00586A FB2D 2A 95 FAC4         BPL    GETIT    IF KEY; GO GET IT
00587A FB2F 32                 PULA
00588A FB30 32                 PULA            ADJUST STACK TO RETURN TO 'PUT'
00589A FB31 39                 RTS             **RETURN**
00590                        * TO ROUTINE THAT CALLED ROUTINE THAT CALLED KEYTST!!!


00592                        *
00593                        * OFFSET ROUTINE
00594                        *
00595A FB32 7D 8137   A OFFSET TST    INIT2    FIRST TIME THRU?
00596A FB35 26 77 FBAE         BNE    OFST     NO!
00597A FB37 7C 8137   A        INC    INIT2    NO MORE OF THIS
00598A FB3A 7F 8134   A        CLR    ROLFLG   PREPARE TO DISPLAY HEX CHARACTERS
00599A FB3D 7F 8135   A        CLR    FLG24
00600A FB40 86 7F     A        LDAA   #%01111111
00601A FB42 20 75 FBB9         BRA    DISP1


00603                        *
00604                        * HEX DIGIT HANDLING ROUTINE
00605                        *
00606A FB44 7D 8134   A ROLENT TST    ROLFLG   CHECK FOR FIRST PASS
00607A FB47 26 0D FB56         BNE    ROLL     IF 0; FIRST PASS
00608A FB49 7C 8134   A        INC    ROLFLG   KNOCK DOWN FIRST PASS FLAG
00609A FB4C 7D 8135   A        TST    FLG24    TEST NIBBLE MODE = 2 OR 4
00610A FB4F 26 12 FB63         BNE    BOT2     IF SET,IN 2 NIBBLE MODE
00611A FB51 7F 8118   A        CLR    HEXBUF   4 NIBBLE MODE
00612A FB54 20 1F FB75         BRA    BOT4
00613A FB56 7D 8135   A ROLL   TST    FLG24    CHECK MODE
00614A FB59 27 0C FB67         BEQ    DUBROL   IF ZERO; GO DO 2-BYTE ROLL
00615A FB5B F6 811A   A        LDAB   HEXBUF+2 GET CURRENT DYSPLAY VALUE
00616A FB5E 58                 ASLB            MOVE LEFT 4 BIT POSITIONS
00617A FB5F 58                 ASLB
00618A FB60 58                 ASLB
00619A FB61 58                 ASLB
00620A FB62 1B                 ABA             ADD NEW NIBBLE (FROM A-REG)
00621A FB63 B7 811A   A BOT2   STAA   HEXBUF+2 UPDATE DISPLAY VALUE
00622A FB66 39                 RTS
00623A FB67 C6 04     A DUBROL LDAB   #4       PREPARE LOOP INDEX
00624A FB69 78 8119   A LOOPM  ASL    HEXBUF+1 LEFT SHIFT WITH MSB TO CARRY
00625A FB6C 79 8118   A        ROL    HEXBUF   LEFT ROLL WITH CARRY TO LSB
00626A FB6F 5A                 DECB
00627A FB70 26 F7 FB69         BNE    LOOPM    GO TILL INDEX (B) GOES TO 0
00628A FB72 BA 8119   A        ORAA   HEXBUF+1 COMBINE NEW NIBBLE
00629A FB75 B7 8119   A BOT4   STAA   HEXBUF+1 UPDATE DISPLAYED VALUE
00630A FB78 39                 RTS             ** RETURN **
```

```
00632                         *
00633                         * KEY FUNCTION DECODING ROUTINE
00634                         *
00635A FB79 8D A7 FB22 FNCDCD BSR    KEYTST   GO TEST FOR A KEY
00636A FB7B 2B 0D FB8A        BMI    HEX      KEY DOWN,SEE IF HEX/FUNCTION
00637A FB7D CE FB9E  A        LDX    #FNHASH  POINT AT LOOK-UP TABLE
00638A FB80 48         HASCLC ASLA            MULTIPLY FUNCTION # BY 2
00639A FB81 BD F94D  A        JSR    ADDXA    ADD TO GET LOOK-UP ADDRESS
00640A FB84 EE 00    A        LDX    0,X      GET HASH TABLE VALUE
00641A FB86 FF 8100  A        STX    MNPTR    SET MAIN POINTER TO NEW ROUTINE
00642A FB89 39                RTS
00643A FB8A 7D 8136  A HEX    TST    FNCFLG   SEE IF IN FSET MODE
00644A FB8D 26 0A FB99        BNE    USERFN   IF SO DO USER'S ROUTINE
00645A FB8F 8D B3 FB44        BSR    ROLENT   SHIFT IN NEW NIBBLE
00646A FB91 8D 78 FC0B        BSR    DYSCOD   CONVERT TO 7-SEG CODES
00647A FB93 86 F0    A DONE   LDAA   #%11110000
00648A FB95 BD F848  A        JSR    CLRDSP
00649A FB98 39                RTS             ** RETURN **
00650A FB99 FE 8102  A USERFN LDX    UHASH    GET USER FUNCTION TABLE ADDRESS
00651A FB9C 20 E2 FB80        BRA    HASCLC   GO CALC HASH ADDR FOR USER FUNCTIO


00653                         *
00654                         * STARTING ADDRESSES OF MAIN ROUTINES
00655                         *
00656A FB9E     FADC   A FNHASH FDB    MEMCH
00657A FBA0     F857   A        FDB    PROMPT
00658A FBA2     F9DD   A        FDB    REGDIS
00659A FBA4     FC64   A        FDB    GO
00660A FBA6     FC4C   A        FDB    FSET
00661A FBA8     FC59   A        FDB    FCLR
00662A FBAA     FE1C   A        FDB    PLOAD
00663A FBAC     FCD0   A        FDB    TBRK
```

```
00665                         *
00666                         * SUBROUTINE TO CALCULATE OFFSET
00667                         *
00668A FBAE BD FB22   A OFST   JSR    KEYTST   OBTAIN A KEY
00669A FBB1 2A 0F FBC2         BPL    FNC      IS IT NOT HEX?
00670A FBB3 8D 8F FB44         BSR    ROLENT   ROLL IN THE HEX CHARACTER
00671A FBB5 8D 54 FC0B DSP     BSR    DYSCOD   ADDR IS DISPLAYED HERE
00672A FBB7 86 70      A       LDAA   #%01110000
00673A FBB9 BD F848   A DISP1  JSR    CLRDSP
00674A FBBC 86 77      A       LDAA   #$77     DISPLAY AN 'A' FOR ADDRESS
00675A FBBE B7 8123   A        STAA   DISBUF+7 PUT IT ON THE DISPLAY
00676A FBC1 39                 RTS             **RETURN**
00677A FBC2 81 03      A FNC   CMPA   #3       A GO KEY?
00678A FBC4 26 44 FC0A         BNE    ENDOFF   TRY AGAIN
00679A FBC6 FE 8111   A        LDX    XTMP3    GET ADDR TO OFFSET FROM(USER PC)
00680A FBC9 08                 INX             ADJUST OFFSET TO ADDR. AFTER
00681A FBCA FF 8111   A        STX    XTMP3    SAVE IT FOR CALCULATIONS
00682A FBCD 09                 DEX             RESTORE TO CORRECT ADDR.
00683A FBCE F6 8119   A        LDAB   HEXBUF+1 LOAD ADDR FROM DISPLAY
00684A FBD1 F0 8112   A        SUBB   XTMP3+1  SUBTRACT USER PC
00685A FBD4 B6 8118   A        LDAA   HEXBUF
00686A FBD7 B2 8111   A        SBCA   XTMP3
00687A FBDA 29 13 FBEF         BVS    BADOFS   OFFSET TOO LARGE
00688A FBDC 27 07 FBE5         BEQ    GOODOF   JUST RIGHT!
00689A FBDE 43                 COMA
00690A FBDF 26 0E FBEF         BNE    BADOFS   GOOD TRY ANY WAY
00691A FBE1 C1 80      A       CMPB   #$80     MAKE SURE OFFSET WITHIN LIMITS
00692A FBE3 20 01 FBE6         BRA    GOOD
00693A FBE5 5D          GOODOF TSTB
00694A FBE6 2B 07 FBEF GOOD    BMI    BADOFS
00695A FBE8 E7 00      A       STAB   0,X
00696A FBEA F7 811A   A        STAB   HEXBUF+2
00697A FBED 20 05 FBF4         BRA    PAST
00698A FBEF C6 FF      A BADOFS LDAB  #$FF
00699A FBF1 F7 811A   A        STAB   HEXBUF+2
00700A FBF4 FF 8118   A PAST   STX    HEXBUF
00701A FBF7 FF 8111   A        STX    XTMP3    RESTORE USER PROGRAM COUNTER
00702A FBFA E6 00      A       LDAB   0,X      GET OLD/CHANGED BYTE
00703A FBFC F7 811B   A        STAB   HEXBUF+3 DISPLAY IT
00704A FBFF 7F 8134   A        CLR    ROLFLG
00705A FC02 7C 8135   A        INC    FLG24    RETURN FLAG TO ORIGINAL STATE
00706A FC05 7F 8137   A        CLR    INIT2
00707A FC08 8D 01 FC0B DYS     BSR    DYSCOD
00708A FC0A 39          ENDOFF RTS             **RETURN**
```

```
                        *
                        * DECODE HEX INTO 7-SEGMENT CODES
                        *
A FC0B 36               DYSCOD PSHA                SAVE A-REG
A FC0C CE 8118   A             LDX    #HEXBUF   POINT AT HEX BUFFER
A FC0F A6 00     A LP01        LDAA   0,X       GET HEX BYTE
A FC11 16                      TAB              MAKE AN EXTRA COPY OF IT
A FC12 54                      LSRB
A FC13 54                      LSRB             RIGHT JUSTIFY HIGH NIBBLE
A FC14 54                      LSRB
A FC15 54                      LSRB             HIGH NIBBLE IN B-REG
A FC16 84 0F     A             ANDA   #$0F      LOW NIBBLE IN A-REG
A FC18 37                      PSHB             SAVE ON STACK
A FC19 36                      PSHA
A FC1A 08                      INX              POINT AT NEXT BYTE
A FC1B 8C 811C   A             CPX    #HEXBUF+4
A FC1E 26 EF FC0F              BNE    LP01      LOOP TILL 4 BYTES CONVERTED
A FC20 CE 8123   A             LDX    #DISBUF+7 POINT AT DISPLAY BUFFER
A FC23 C6 07     A             LDAB   #7        INITIALIZE LOOP INDEX
A FC25 FF 810F   A LP02        STX    XTMP2     SAVE FOR NOW
A FC28 CE FC3C   A             LDX    #DYSTBL   POINT AT LOOK UP TABLE
A FC2B 32                      PULA
A FC2C BD F94D   A             JSR    ADDXA     ADD TO GET LOOK-UP ADDRESS
A FC2F A6 00     A             LDAA   0,X       GET SEVEN-SEGMENT CODE
A FC31 FE 81[illegible]F   A             LDX    XTMP2     RECOVER DYSBUF POINTER
A FC34 A7 00     A             STAA   0,X       STORE CODE TO DYSBUF
A FC36 09                      DEX              POINT TO NEXT DYSBUF POSITION
A FC37 5A                      DECB             DECREMENT LOOP INDEX
A FC38 2A EB FC25              BPL    LP02      LOOP TILL ALL DYSBUF CODED
A FC3A 32                      PULA             RESTORE A-REG
A FC3B 39                      RTS              ** RETURN **


                        *
                        * 7-SEGMENT LOOK-UP TABLE
                        *
A FC3C      3F   A DYSTBL FCB    $3F       '0'
A FC3D      06   A        FCB    $06       '1'
A FC3E      5B   A        FCB    [illegible]  '2'
A FC3F      4F   A        FCB    $4F       '3'
A FC40      66   A        FCB    $66       '4'
A FC41      6D   A        FCB    $6D       '5'
A FC42      7D   A        FCB    $7D       '6'
A FC43      07   A        FCB    $07       '7'
A FC44      7F   A        FCB    $7F       '8'
A FC45      67   A        FCB    $67       '9'
A FC46      77   A        FCB    $77       'A'
A FC47      7C   A        FCB    $7C       'B'
A FC48      39   A        FCB    $39       'C'
A FC49      5E   A        FCB    $5E       'D'
A FC4A      79   A        FCB    $79       'E'
A FC4B      71   A        FCB    $71       'F'
```

```
                         *
                         * SPECIAL FUNCTION SET
                         *
FC4C CE 716D  A FSET   LDX    #$716D   7-SEG CODE FOR 'FS'
FC4F FF 8122  A        STX    DISBUF+6 PUT IT IN 7TH & 8TH DIGITS
FC52 86 01    A        LDAA   #1
FC54 B7 8136  A        STAA   FNCFLG   SET FUNCTION FLAG
FC57 20 08 FC61        BRA    ALDUN    EXIT
```

```
                         *
                         * SPECIAL FUNCTION CLEAR
                         *
FC59 86 C0    A FCLR   LDAA   #%11000000
FC5B BD F848  A        JSR    CLRDSP
FC5E 7F 8136  A        CLR    FNCFLG   CLEAR FUNCTION FLAG
FC61 7E F871  A ALDUN  JMP    PROMP1   ** EXIT (SEE NOTE)**
                * EXITS TO PROMPT AFTER FLAG INITIALIZATION
                * SO AS NOT TO DISTURB ANY OPERATION IN PROGRESS
                * --RESULTS IN FUNCTION DECODE ADDRESS BEING PUT AT
                * MNPTR SO THAT FUNCTION TAKES CONTROL
```

```
                     ******************************************************
                     *
                     * GO TO ROUTINE
                     *
                     ******************************************************
A FC64 BD F839  A GO      JSR    EFPI      ENABLE FRONT PANEL NMI
A FC67 0D                 SEC
A FC68 76 8130  A         ROR    EXFLAG    MAKE EXFLAG NEGATIVE
A FC6B 7D 8134  A         TST    ROLFLG    WAS NUMERIC DATA INPUT BEFORE "G"?
A FC6E 27 06 FC76         BEQ    FTST      NO,BYPASS UPC INITIALIZATION
A FC70 FE 8118  A         LDX    HEXBUF    YES,GET USER PROG COUNTER
A FC73 FF 8129  A         STX    UPC       INITIALIZE USER PROG COUNTER
A FC76 7D 8136  A FTST    TST    FNCFLG    TEST FOR FUNCTION SET MODE
A FC79 27 0C FC87         BEQ    NORMGO    CLEAR,GO TO NORMAL GO ENTRY
A FC7B FE 8129  A         LDX    UPC       GET ADDRESS OF USER MAIN PROG
A FC7E FF 8100  A         STX    MNPTR     INSTALL IT IN"MAIN"POINTER
A FC81 CE F90B  A         LDX    #PUT
A FC84 FF 8129  A         STX    UPC       MAKE ROUTINE JUMP TO"PUT"AFTER SP
A FC87 BD FDE1  A NORMGO  JSR    INBKPT    INSTALL BREAKPOINTS
A FC8A BE 812B  A GOTO    LDS    USP       LOAD USER'S STACK POINTER
A FC8D 86 55    A         LDAA   #$55      BEGIN TEST FOR EXISTENCE OF STACK
A FC8F 36                 PSHA             TRY TO PUT ON USER'S STACK
A FC90 32                 PULA             IF HIS SP WAS OK A-REG=$55
A FC91 81 55    A         CMPA   #$55      SEE IF IT IS
A FC93 26 10 FCA5         BNE    BADSTK    IF NOT; GO CANNOT CONTINUE
A FC95 B6 812A  A         LDAA   UPC+1     GET LOW BYTE OF USER "PC"
A FC98 36                 PSHA             STACK IT TO PREPARE FOR RTS
A FC99 B6 8129  A         LDAA   UPC       HIGH BYTE
A FC9C 36                 PSHA             STACK IT
A FC9D 86 AA    A         LDAA   #$AA      TEST PATTERN FOR STACK TEST
A FC9F 36                 PSHA
A FCA0 32                 PULA             IF OK IT WILL BE $AA
A FCA1 81 AA    A         CMPA   #$AA      SEE IF IT IS
A FCA3 27 11 FCB6         BEQ    GOEXIT    YES, CONTINUE TO LOAD USER REGS
A FCA5 BD F846  A BADSTK  JSR    CLRD      CLEAR DISPLAY
A FCA8 CE 6D73  A         LDX    #$6D73    7-SEG FOR 'SP'
A FCAB FF 811E  A         STX    DISBUF+2
A FCAE 86 53    A         LDAA   #$53      7-SEG FOR '?'
A FCB0 B7 8123  A         STAA   DISBUF+7
A FCB3 7E F861  A         JMP    PROMPG    ENTER PROMPT SEQ AFTER DISBUF INIT
A FCB6 FE 8127  A GOEXIT  LDX    UX        LOAD USER X-REG
A FCB9 F6 8125  A         LDAB   UB        LOAD USER B-REG
A FCBC B6 8126  A         LDAA   UA        GET USER A-REG
A FCBF 36                 PSHA             SAVE ON STACK (USER'S STACK)
A FCC0 7D 812F  A         TST    NFLAG
A FCC3 27 05 FCCA         BEQ    ARN17     IF NOT TRACE MODE BYPASS TIMER INI
A FCC5 86 B2    A         LDAA   #$B2      SET-UP TIMER FOR TRACE TRIGGER
A FCC7 B7 8085  A         STAA   TCNTRL
A FCCA B6 8124  A ARN17   LDAA   UCC       PREPARE TO INITIALIZE USER COND. C
A FCCD 06                 TAP              GET THEM INTO CC-REG
A FCCE 32                 PULA             LOAD USER A-REG WITHOUT AFFECTING
A FCCF 39                 RTS              **EXIT TO USER PROG**
```

```
                        ************************************************************
                        *
                        *       BREAK POINT SUBROUTINE
                        *
                        ************************************************************
FCD0 7D 8136   A TBRK   TST     FNCFLG
FCD3 26 05 FCDA         BNE     CONT
FCD5 7C 812F   A TRACE  INC     NFLAG
FCD8 20 B0 FC8A         BRA     GOTO
FCDA 7D 812D   A CONT   TST     INIT1
FCDD 26 17 FCF6         BNE     ACTION   1ST TIME THRU?
FCDF 7F 8134   A        CLR     ROLFLG
FCE2 7C 812D   A        INC     INIT1    DON'T WANT TO DO THIS AGAIN
FCE5 7F 8142   A        CLR     NUMBER   NO BKPTS.
FCE8 CE 8143   A        LDX     #BOF     GET TABLE ADDR. OF 1ST BPT.
FCEB FF 813E   A        STX     POINTR   SET UP TABLE POINTER
FCEE FF 8140   A        STX     TOFT     INITIALIZE TOP OF TABLE POINTER
FCF1 7F 811B   A        CLR     HEXBUF+3 SHOW WHICH BPT IS DISPLAYED
FCF4 20 7D FD73         BRA     DISPLY   DO THE ACTUAL LED DISPLAY
FCF6 7D 8137   A ACTION TST     INIT2
FCF9 26 0E FD09         BNE     ACTON2
FCFB 7C 8137   A        INC     INIT2
FCFE FE 8143   A        LDX     BOF
FD01 F6 8142   A        LDAB    NUMBER
FD04 F7 811B   A        STAB    HEXBUF+3
FD07 20 73 FD7C         BRA     DISPY1
FD09 BD FB22   A ACTON2 JSR     KEYTST   WHICH KEY WAS PRESSED
FD0C 2A 08 FD16         BPL     FUNCTN   DECODE A FUNCTION KEY
FD0E 7F 8135   A        CLR     FLG24    HANDLE HEX KEYS-SET UP DISPLAY
FD11 BD FB44   A        JSR     ROLENT   ROLL HEX NIBBLE INTO HEXBUF
FD14 20 69 FD7F         BRA     DISPY    GO DISPLAY IT
FD16 FE 813E   A FUNCTN LDX     POINTR   SET INDEX TO TABLE POINTER
FD19 81 03     A        CMPA    #$3
FD1B 26 24 FD41         BNE     FSET33   TRY ADDING NEW BKPT.
FD1D 8C 8143   A        CPX     #BOF
FD20 26 05 FD27         BNE     SKIP2    NOT THE BOTTOM OF TABLE
FD22 BC 8140   A        CPX     TOFT     IS IT ALSO THE TOP OF TABLE
FD25 27 0A FD31         BEQ     SKIP     DON'T MESS WITH IT
FD27 08          SKIP2  INX
FD28 08                 INX
FD29 BC 8140   A        CPX     TOFT     SEEK TOP OF TABLE
FD2C 26 03 FD31         BNE     SKIP
FD2E CE 8143   A        LDX     #BOF     YES WRAP AROUND
FD31 FF 813E   A SKIP   STX     POINTR   UPDATE TABLE POINTR
FD34 EE 00     A        LDX     0,X      LOA BPT. ADDR.
FD36 FF 8118   A        STX     HEXBUF   PUT  IN DISPLAY
FD39 B6 8142   A        LDAA    NUMBER   LOAD #OF BKPTS.
FD3C B7 811B   A        STAA    HEXBUF+3
FD3F 20 3E FD7F         BRA     DISPY    DISPLAY IT
```

```
A FD41 81 04      A FSET33 CMPA   #4         ADD NEW BPT.
A FD43 26 68 FDAD          BNE    FCLR33     NO MAYBE REMOVE ONE
A FD45 B6 8142    A        LDAA   NUMBER
A FD48 81 08      A        CMPA   #8         NO MORE ROOM?
A FD4A 2C 3B FD87          BGE    ENDBPT     YES GET OUT
A FD4C 7F 8134    A        CLR    ROLFLG
A FD4F 8D 37 FD88          BSR    FNDBRK
A FD51 27 34 FD87          BEQ    ENDBPT     DON'T ADD THE SAME BREAKPOINT
A FD53 FE 8140    A        LDX    TOFT
A FD56 B6 8118    A        LDAA   HEXBUF
A FD59 A7 00      A        STAA   0,X
A FD5B B6 8119    A        LDAA   HEXBUF+1
A FD5E A7 01      A        STAA   1,X        ADD NEW BKPT.
A FD60 FF 813E    A        STX    POINTR
A FD63 08                  INX
A FD64 08                  INX               ADVANCE INDEX
A FD65 FF 8140    A        STX    TOFT       UPDATE TOP OF TABLE POINTER
A FD68 7C 8142    A        INC    NUMBER     UPDATE # OF BKPTS.
A FD6B B6 8142    A        LDAA   NUMBER
A FD6E B7 811B    A        STAA   HEXBUF+3 DISPLAY UPDATE
A FD71 20 0C FD7F          BRA    DISPY
A FD73 7F 8143    A DISPLY CLR    BOF
A FD76 7F 8144    A        CLR    BOF+1
A FD79 CE 0000    A        LDX    #0
A FD7C FF 8118    A DISPY1 STX    HEXBUF
A FD7F BD FC0B    A DISPY  JSR    DYSCOD     DISPLAY BKPT
A FD82 86 70      A        LDAA   #%01110000
A FD84 BD F848    A        JSR    CLRDSP
A FD87 39           ENDBPT RTS               **RETURN**
A FD88 CE 8143    A FNDBRK LDX    #BOF
A FD8B FF 813E    A        STX    POINTR
A FD8E B6 8142    A        LDAA   NUMBER     TOP OF TABLE=BEGINNING OF TABLE
A FD91 27 17 FDAA          BEQ    NOTFND     DO NOTHING
A FD93 B6 8118    A        LDAA   HEXBUF     GET BKPT TO REMOVE
A FD96 F6 8119    A        LDAB   HEXBUF+1 BOTH BYTES ARE NEEDED
A FD99 A1 00      A LOOP1M CMPA   0,X        SEARCH FOR BPT TO REMOVE
A FD9B 26 04 FDA1          BNE    INCX       NOT FOUND
A FD9D E1 01      A        CMPB   1,X        CONT. SEARCH
A FD9F 27 0B FDAC          BEQ    FOUND      FOUND IT!
A FDA1 BC 8140    A INCX   CPX    TOFT       TOP OF TABLE?
A FDA4 27 04 FDAA          BEQ    NOTFND
A FDA6 08                  INX
A FDA7 08                  INX
A FDA8 20 EF FD99          BRA    LOOP1M
A FDAA 86 01      A NOTFND LDAA   #1
A FDAC 39           FOUND  RTS               **RETURN**
```

```
00934A FDAD 81 05      A FCLR33 CMPA    #5         CLEAR A BKPT?
00935A FDAF 26 D6 FD87          BNE     ENDBPT
00936A FDB1 8D D5 FD88          BSR     FNDBRK
00937A FDB3 26 D2 FD87          BNE     ENDBPT
00938A FDB5 7F 8134    A        CLR     ROLFLG
00939A FDB8 BC 8140    A LOOP20 CPX     TOFT
00940A FDBB 27 0C FDC9          BEQ     OUT
00941A FDBD A6 02      A        LDAA    2,X        MOVE TABLE DOWN
00942A FDBF A7 00      A        STAA    0,X
00943A FDC1 A6 03      A        LDAA    3,X
00944A FDC3 A7 01      A        STAA    1,X
00945A FDC5 08                  INX
00946A FDC6 08                  INX                MOVE DOWN NEXT BKPT
00947A FDC7 20 EF FDB8          BRA     LOOP20     REACH TOP OF TABLE?
00948A FDC9 FE 8140    A OUT    LDX     TOFT
00949A FDCC 09                  DEX
00950A FDCD 09                  DEX                DEC TO SHOW FEWER BKPTS
00951A FDCE FF 8140    A        STX     TOFT
00952A FDD1 7A 8142    A        DEC     NUMBER     DISPLAY NEW NUMBER-EXIT
00953A FDD4 B6 8142    A        LDAA    NUMBER
00954A FDD7 B7 811B    A        STAA    HEXBUF+3
00955A FDDA 27 97 FD73          BEQ     DISPLY
00956A FDDC FE 8143    A        LDX     BOF        GET FIRST BKPT
00957A FDDF 20 9B FD7C          BRA     DISPY1
00958A FDE1 CE 8143    A INBKPT LDX     #BOF       BEGINNING OF TABLE
00959A FDE4 C6 3F      A        LDAB    #$3F       BKPT OPCODE
00960A FDE6 BC 8140    A LOOP   CPX     TOFT
00961A FDE9 27 30 FE1B          BEQ     END
00962A FDEB FF 813E    A        STX     POINTR     KEEP OLD BKPT ADDR.
00963A FDEE EE 00      A        LDX     0,X        READ ACTUAL BKPT ADDR.
00964A FDF0 A6 00      A        LDAA    0,X        GET OPCODE
00965A FDF2 E7 00      A        STAB    0,X        PLACE INTERRUPT IN ITS PLACE
00966A FDF4 FE 813E    A        LDX     POINTR     RESTORE OLD BKPT TABLE ADDR.
00967A FDF7 A7 10      A        STAA    16,X       SAVE OPCODE ON HASHED TABLE
00968A FDF9 08                  INX
00969A FDFA 08                  INX                NEXT BKPT
00970A FDFB 20 E9 FDE6          BRA     LOOP       CONTINUE TILL ALL BKPTS INSERTED
00971A FDFD CE 8143    A RMBKPT LDX     #BOF       BEGINNING OF BKPT TABLE
00972A FE00 BC 8140    A LOOP40 CPX     TOFT
00973A FE03 27 16 FE1B          BEQ     END
00974A FE05 FF 813E    A        STX     POINTR     USE POINTR AS TEMP STORAGE
00975A FE08 E6 10      A        LDAB    16,X       RECOVER OPCODE
00976A FE0A EE 00      A        LDX     0,X        RETRIEVE OPCODE'S ADDR.
00977A FE0C A6 00      A        LDAA    0,X        CHECK FOR BKPT
00978A FE0E 81 3F      A        CMPA    #$3F
00979A FE10 26 02 FE14          BNE     WHOOPS     ITS CHANGED DON'T CHANGE AGAIN
00980                  *        THIS ALLOWS USER TO CHANGE BREAKPOINTS IF HE WISHE
00981A FE12 E7 00      A        STAB    0,X        REPLACE OPCODE
00982A FE14 FE 813E    A WHOOPS LDX     POINTR     GET BACK TABLE ADDR
00983A FE17 08                  INX
00984A FE18 08                  INX                MOVE ON TO NEXT BKPT
00985A FE19 20 E5 FE00          BRA     LOOP40     YES FINISH UP
00986A FE1B 39           END    RTS                **RETURN**
```

```
                        *
                        * SET UP PUNCH/LOAD
                        *
FE1C 7F 8132  A PLOAD   CLR    VERIFY
FE1F C6 03    A         LDAB   #3        INITIALIZE ACIA
FE21 F7 800A  A         STAB   ACIA0
FE24 FE 8111  A         LDX    XTMP3
FE27 7D 8136  A         TST    FNCFLG
FE2A 26 75 FEA1         BNE    LD
FE2C 7D 8137  A PNCH    TST    INIT2
FE2F 26 10 FE41         BNE    INPUT
FE31 7F 8131  A         CLR    S3FLAG
FE34 7F 8135  A         CLR    FLG24
FE37 7C 8137  A         INC    INIT2     LAST TIME THRU
FE3A C6 0B    A         LDAB   #$0B      PUT A PROMPT FOR BEG. ADDR.
FE3C CE 813A  A         LDX    #BEGA
FE3F 20 25 FE66         BRA    DISP2
FE41 BD FB22  A INPUT   JSR    KEYTST    BEGIN KEYBOARD HANDLER
FE44 2A 05 FE4B         BPL    GO00      NOT A HEX KEY
FE46 BD FB44  A         JSR    ROLENT    ROLL HEX # IN
FE49 20 28 FE73         BRA    DISP11    GO DISPLAY IT
FE4B 8D 32 FE7F GO00    BSR    FSTCLR    DECODE KEY COMMAND
FE4D F6 8118  A         LDAB   HEXBUF    RECOVER INPUTTED ADDR.
FE50 E7 00    A         STAB   0,X
FE52 F6 8119  A         LDAB   HEXBUF+1
FE55 E7 01    A         STAB   1,X
FE57 7D 8137  A         TST    INIT2
FE5A 2B 20 FE7C         BMI    PUNCH1    GO PUNCH INSTEAD
FE5C C6 80    A         LDAB   #$80
FE5E F7 8137  A         STAB   INIT2     KEEP FROM DOING THIS AGAIN
FE61 CE 813C  A         LDX    #ENDA
FE64 C6 0E    A         LDAB   #$0E      PUT END ADDR. PROMPT UP
FE66 FF 8111  A DISP2   STX    XTMP3
FE69 7F 8134  A         CLR    ROLFLG
FE6C F7 811B  A         STAB   HEXBUF+3
FE6F 86 7F    A         LDAA   #%01111111
FE71 20 02 FE75         BRA    DISP10
FE73 86 70    A DISP11  LDAA   #%01110000
FE75 BD FC0B  A DISP10  JSR    DYSCOD    DEPENDS ON DYSCOD NOT CHANGING A
FE78 BD F848  A         JSR    CLRDSP
FE7B 39         PEND    RTS              **RETURN**
FE7C 7E FFA7  A PUNCH1  JMP    PUNCH
FE7F 26 07 FE88 FSTCLR  BNE    NXT0      VERIFY?
FE81 7C 8132  A         INC    VERIFY    SET VERIFY MODE FLAG
FE84 32                 PULA             DO NOT WISH TO RETURN
FE85 32                 PULA
FE86 20 38 FEC0         BRA    GO1       INSTEAD DO A TAPE VERIFY
FE88 81 04    A NXT0    CMPA   #4        SET S3FLAG?
FE8A 26 05 FE91         BNE    NXT       NO
FE8C 7C 8131  A         INC    S3FLAG    GO INTO 300 BAUD MODE
FE8F 20 0D FE9E         BRA    RET44
```

```
01041A FE91 81 05     A NXT     CMPA    #5         CLEAR S3FLAG?
01042A FE93 26 05 FE9A          BNE     NXT1       NO
01043A FE95 7F 8131  A          CLR     S3FLAG
01044A FE98 20 04 FE9E          BRA     RET44
01045A FE9A 81 03     A NXT1    CMPA    #3         PUNCH/LOAD?
01046A FE9C 27 02 FEA0          BEQ     ENDFST     YES
01047A FE9E 32          RET44   PULA               GO BACK TO PUT
01048A FE9F 32                  PULA
01049A FEA0 39          ENDFST  RTS                **RETURN**


01051                   *
01052                   * LOAD 1200 BAUD
01053                   *
01054A FEA1 BD F846  A LD       JSR     CLRD
01055A FEA4 7D 8133  A          TST     ERROR
01056A FEA7 26 47 FEF0          [redacted]    MESAGE
01057A FEA9 86 4F     A         LDAA    #$4F       INITIALIZE DISPLAY
01058A FEAB B7 811F  A          STAA    DISBUF+3
01059A FEAE CE 065B  A          LDX     #$065B
01060A FEB1 FF 811C  A KEYM     STX     DISBUF
01061A FEB4 86 53     A         LDAA    #$53
01062A FEB6 B7 8123  A          STAA    DISBUF+7
01063A FEB9 BD FB22  A          JSR     KEYTST     START KEY HANDLER
01064A FEBC 2B 30 FEEE          BMI     QUIT
01065A FEBE 8D BF FE7F          BSR     FSTCLR
01066A FEC0 7F 8133  A GO1      CLR     ERROR
01067A FEC3 B7 8138  A          STAA    LDFLAG     SET LDFLAG TO OTHER THAN 0
01068A FEC6 86 11     A         LDAA    #%00010001
01069A FEC8 B7 800A  A          STAA    ACIA0
01070A FECB BD F846  A          JSR     CLRD
01071A FECE 7D 8131  A          TST     S3FLAG
01072A FED1 26 22 FEF5          BNE     S300LD     GO TO 300 BAUD ROUTINE
01073          FED3  A LOAD     EQU     *
01074A FED3 8D 73 FF48 L1       BSR     INCHR
01075A FED5 C1 96     A         CMPB    #$96       LOOK FOR SYNC
01076A FED7 26 FA FED3          BNE     L1
01077A FED9 4F          L2      CLRA               CLEAR CHECKSUM
01078A FEDA 8D 6C FF48          BSR     INCHR      GET BYTE COUNT
01079A FEDC 27 10 FEEE          BEQ     QUIT       QUIT IF ZERO
01080A FEDE 8D 2B FF0B          BSR     PIN
01081A FEE0 8D 66 FF48          BSR     INCHR      GET CHECKSUM
01082A FEE2 4C                  INCA               TEST FOR CORRECTNESS
01083A FEE3 26 06 FEEB          BNE     ERR        OOPS!
01084A FEE5 8D 61 FF48          BSR     INCHR      NEXT BLOCK OF DATA
01085A FEE7 C1 96     A         CMPB    #$96       ANOTHER SYNC?
01086A FEE9 27 EE FED9          BEQ     L2         YES, GO GET IT
01087A FEEB 7C 8133  A ERR      INC     ERROR
01088A FEEE 20 83 FE73 QUIT     BRA     DISP11
01089A FEF0 CE 396D  A MESAGE   LDX     #$396D
01090A FEF3 20 BC FEB1          BRA     KEYM
```

```
01092                        *
01093                        * 300 BAUD LOAD ROUTINE FOR TAPE
01094                        *
01095A FEF5 8D 51 FF48 S300LD BSR    INCHR
01096A FEF7 C1 42    A        CMPB   #'B      START OF BINARY?
01097A FEF9 27 09 FF04        BEQ    RDBLCK   YES
01098A FEFB C1 47    A        CMPB   #'G      END OF FILE?
01099A FEFD 26 F6 FEF5        BNE    S300LD
01100A FEFF 7F 8131  A        CLR    S3FLAG
01101A FF02 20 EA FEEE        BRA    QUIT
01102A FF04 8D 42 FF48 RDBLCK BSR    INCHR    GET BYTE COUNT
01103A FF06 5C                INCB            SAVE IT
01104A FF07 8D 02 FF0B        BSR    PIN
01105A FF09 20 EA FEF5        BRA    S300LD


01107                        *
01108                        * INPUT ROUTINE FOR LD AND S300LD
01109                        *
01110A FF0B CE 8139  A PIN    LDX    #LDTBLE
01111A FF0E E7 00    A        STAB   0,X
01112A FF10 8D 36 FF48        BSR    INCHR    GET MSB ADDR.
01113A FF12 E7 01    A        STAB   1,X
01114A FF14 8D 32 FF48        BSR    INCHR    GET LSB ADDR.
01115A FF16 E7 02    A        STAB   2,X
01116A FF18 EE 01    A        LDX    1,X      RETRIEVE START ADDR.
01117A FF1A 7D 8137  A        TST    INIT2
01118A FF1D 26 06 FF25        BNE    NEXT1
01119A FF1F 7C 8137  A        INC    INIT2
01120A FF22 FF 8129  A        STX    UPC      SAVE INITIAL STARTING ADDR.
01121A FF25 8D 21 FF48 NEXT1  BSR    INCHR    GET DATA
01122A FF27 7D 8132  A        TST    VERIFY   VERIFY MODE?
01123A FF2A 27 08 FF34        BEQ    STORE    NO, SO STORE CHAR IN MEMORY
01124A FF2C E1 00    A        CMPB   0,X      TEST AGAINST WHATS IN MEMORY
01125A FF2E 27 06 FF36        BEQ    NEXT2    ITS THE SAME-CHECK NEXT ONE
01126A FF30 32                PULA            BRANCHING INSTEAD OF RETURNING
01127A FF31 32                PULA
01128A FF32 20 B7 FEEB        BRA    ERR      PRINT ERROR MESSAGE
01129A FF34 E7 00    A STORE  STAB   0,X      PUT IT IN MEMORY
01130A FF36 08         NEXT2  INX
01131A FF37 7A 8139  A        DEC    LDTBLE   DEC BYTE COUNT
01132A FF3A 26 E9 FF25        BNE    NEXT1    AGAIN?
01133A FF3C 39                RTS             **RETURN**
```

```
01135                    *
01136                    * PUNCH LEADER
01137                    *
01138A FF3D CE 0019  A PNLDR1 LDX    #$0019
01139A FF40 C6 FF    A PNLDR  LDAB   #$FF      OUTPUT ALL ONES
01140A FF42 8D 28 FF6C        BSR    WRITE
01141A FF44 09                DEX
01142A FF45 26 F9 FF40        BNE    PNLDR
01143A FF47 39                RTS              **RETURN**



01145                    *
01146                    * SUBROUTINE TO PUNCH AND LOAD A BYTE
01147                    *
01148A FF48 36           INCHR  PSHA             INPUT A CHARACTER
01149A FF49 B6 800A  A INCHR2 LDAA   ACIA0     LOOK FOR A NEW CHARACTER
01150A FF4C 47                ASRA             TEST FOR ONE
01151A FF4D 24 FA FF49        BCC    INCHR2    NOT THERE YET
01152A FF4F 32           RQ     PULA
01153A FF50 BD FFE4  A        JSR    IRQ
01154A FF53 39                RTS              **RETURN**



01156                    *
01157                    * OUTPUT ROUTINE FOR S1200P AND S300P
01158                    *
01159A FF54 E6 01    A POUT   LDAB   1,X       OUTPUT MSB ADDR.
01160A FF56 8D 14 FF6C        BSR    WRITE
01161A FF58 E6 02    A        LDAB   2,X       OUTPUT LSB ADDR.
01162A FF5A 8D 10 FF6C        BSR    WRITE
01163A FF5C EE 01    A        LDX    1,X       SET INDEX TO DATA
01164A FF5E E6 00    A P2     LDAB   0,X       OUTPUT DATA BYTE
01165A FF60 8D 0A FF6C        BSR    WRITE
01166A FF62 08                INX
01167A FF63 7A 8139  A        DEC    BCOUNT
01168A FF66 26 F6 FF5E        BNE    P2
01169A FF68 FF 813A  A        STX    BEGA      UPDATE BEGINNING ADDR
01170A FF6B 39                RTS
01171A FF6C 36           WRITE  PSHA
01172A FF6D 7F 8138  A        CLR    LDFLAG
01173A FF70 B6 800A  A WRITE2 LDAA   ACIA0
01174A FF73 47                ASRA
01175A FF74 47                ASRA
01176A FF75 24 F9 FF70        BCC    WRITE2
01177A FF77 20 D6 FF4F        BRA    RQ
```

```
                          *
                          * 300 BAUD PUNCH ROUTINE
                          *
FF79 CE 0200   A S300P  LDX     #$200
FF7C 8D C2 FF40         BSR     PNLDR     PUNCH LEADER
FF7E 8D 58 FFD8 PUND10  BSR     ADDR      DETERMINE MAX BLOCK LENGTH
FF80 27 02 FF84         BEQ     PUND25    DIFF LESS THAN 255
FF82 C6 FF     A        LDAB    #$FF      YES,SET BLOCK = 255
FF84 E7 00     A PUND25 STAB    0,X       SAVE BYTE COUNT
FF86 C6 42     A        LDAB    #'B       PUNCH B
FF88 8D E2 FF6C         BSR     WRITE
FF8A E6 00     A        LDAB    0,X       PUNCH BYTE COUNT
FF8C 8D DE FF6C         BSR     WRITE
FF8E 6C 00     A        INC     0,X       ADJUST BYTE COUNT
FF90 8D C2 FF54         BSR     POUT
FF92 8D A9 FF3D         BSR     PNLDR1    PUNCH 25 ONES
FF94 FE 813A   A        LDX     BEGA      RESTORE XR
FF97 09                 DEX
FF98 BC 813C   A        CPX     ENDA
FF9B 26 E1 FF7E         BNE     PUND10    NO
FF9D C6 47     A        LDAB    #'G
FF9F 8D CB FF6C         BSR     WRITE
FFA1 7F 8137   A QUIT1  CLR     INIT2
FFA4 8D 97 FF3D         BSR     PNLDR1
FFA6 39                 RTS               **RETURN**


                          *
                          * PUNCH EITHER 300 OR 1200 BAUD
                          *
FFA7 86 51     A PUNCH  LDAA    #%01010001
FFA9 B7 800A   A        STAA    ACIA0
FFAC 7D 8131   A        TST     S3FLAG
FFAF 26 C8 FF79         BNE     S300P     GO PUNCH A 300 BAUD TAPE
```

```
01213                       *
01214                       * PUNCH 1200 BAUD BINARY
01215                       *
01216A FFB1 CE 0900   A S1200P LDX    #$900
01217A FFB4 8D 8A FF40         BSR    PNLDR
01218A FFB6 FE 813C   A        LDX    ENDA      RETREIVE END ADDR.
01219A FFB9 [redacted]         INX              MAKE IT ONE MORE
01220A FFBA FF 813C   A        STX    ENDA
01221A FFBD 8D 19 FFD8 LOOP00 BSR    ADDR      CALCULATE BLOCK LENGTH
01222A FFBF 27 02 FFC3         BEQ    P1
01223A FFC1 C6 FF     A        LDAB   #$FF      DEFAULT BYTE COUNT
01224A FFC3 E7 00     A P1     STAB   0,X       BYTE COUNT SAVED
01225A FFC5 C6 96     A        LDAB   #$96      OUTPUT SYNC
01226A FFC7 8D A3 FF6C         BSR    WRITE
01227A FFC9 4F                 CLRA             CLEAR CHECKSUM
01228A FFCA E6 00     A        LDAB   0,X       OUTPUT BLOCKLENGTH
01229A FFCC 8D 9E FF6C         BSR    WRITE
01230A FFCE 27 D1 FFA1         BEQ    QUIT1     GET OUT IF ZERO LENGTH
01231A FFD0 8D 82 FF54         BSR    POUT
01232A FFD2 43                 COMA             OUTPUT CHECKSUM
01233A FFD3 16                 TAB
01234A FFD4 8D 96 FF6C         BSR    WRITE
01235A FFD6 20 E5 FFBD         BRA    LOOP00    DO NEXT BLOCK


01237                       *
01238                       * ADDR IS USED TO CALCULATE BLOCK SIZE IN LOAD
01239                       *
01240A FFD8 CE 8139   A ADDR   LDX    #LDTBLE
01241A FFDB E6 04     A        LDAB   4,X       LSB END ADDR.
01242A FFDD E0 02     A        SUBB   2,X       LSB START ADDR.
01243A FFDF A6 03     A        LDAA   3,X       MSB END ADDR.
01244A FFE1 A2 01     A        SBCA   1,X       MSB START ADDR.
01245A FFE3 39                 RTS
01246A FFE4 7D 8138   A IRQ    TST    LDFLAG    DO THE ACTUAL OUTPUT/INPUT
01247A FFE7 26 05 FFEE         BNE    INPUT1
01248A FFE9 F7 800B   A        STAB   ACIA1
01249A FFEC 20 03 FFF1         BRA    IRQEND
01250A FFEE F6 800B   A INPUT1 LDAB   ACIA1
01251A FFF1 1B          IRQEND ABA              CALCULATE CHECKSUM
01252A FFF2 39                 RTS              **RETURN**


01254A FFF8                    ORG    $FFF8
01255A FFF8    F95F   A IRQV   FDB    UIRQ      IRQ VECTOR
01256A FFFA    F8B3   A SWIV   FDB    SWINT     SWI VECTOR
01257A FFFC    F87F   A NMIV   FDB    NONMSK    NMI VECTOR
01258A FFFE    F800   A RSTV   FDB    RESTRT    RESTART VECTOR
```

```
01260                    * D3BUG OPERATING STACK
01261                    *
01262A 8100                    ORG     $8100
01263A 8100      0002  A MNPTR  RMB     2         ADDRESS OF 'MAIN' LOOP
01264A 8102      [redacted]  A UHASH  RMB     2         USER SPECIAL FUNCTION POINTER
01265A 8104      0002  A UNMIV  RMB     2         USER NMI VECTOR
01266A 8106      0002  A UIRQV  RMB     2         USER IRQ VECTOR
01267A 8108      0002  A USWIV  RMB     2         USER SWI VECTOR
01268                    * GTEMP & GXTMP ARE USED IN AN INTERRUPT DRIVEN
01269                    *  ROUTINE -- USE EXTREME CARE IF USED ELSEWHERE
01270A 810A      0001  A GTEMP  RMB     1         TEMP STORAGE LOCATION
01271A 810B      0002  A GXTMP  RMB     2         DOUBLE BYTE TEMP STORAGE
01272A 810D      0002  A XTMP1  RMB     2         .  "    "    "    "
01273A 810F      0002  A XTMP2  RMB     2         .  "    "    "    "
01274A 8111      0002  A XTMP3  RMB     2
01275A 8113      0002  A XCALC  RMB     2         TEMP CALCULATION BUFFER
01276A 8115      0001  A DIGEN  RMB     1         ROTATING DIGIT ENABLE BIT
01277A 8116      0001  A REGNUM RMB     1         REGISTER NUMBER FOR REGDIS
01278A 8117      0001  A KEY    RMB     1         DECODED KEY VALUE
01279A 8118      0004  A HEXBUF RMB     4         4-BYTE HEX BUFFER
01280A 811C      0008  A DISBUF RMB     8         8-BYTE DISPLAY BUFFER (7-SEG)
01281            8124  A SAVSTK EQU     *
01282A 8124      0001  A UCC    RMB     1         USER REGISTER CONDITION CODES
01283A 8125      0001  A UB     RMB     1         ACCUMULATOR B
01284A 8126      0001  A UA     RMB     1         ACCUMULATOR A
01285A 8127      0002  A UX     RMB     2         INDEX REGISTER
01286A 8129      0002  A UPC    RMB     2         PROGRAM COUNTER
01287A 812B      0002  A USP    RMB     2         USER STACK POINTER
01288A 812D      0001  A INIT1  RMB     1         FRESH START FLAG
01289A 812E      0001  A ROWCOL RMB     1         CODED KEY CLOSURE FROM GET
01290A 812F      0001  A NFLAG  RMB     1         TRACE FLAG
01291A 8130      0001  A EXFLAG RMB     1         EXPANSION ROM PRESENCE FLAG
01292A 8131      0001  A S3FLAG RMB     1         USED IN PLOAD FOR SPEED OF I/O
01293            8132  A VERIFY EQU     *         FLAG USED IN PLOAD FOR TAPE VERIFY
01294A 8132      0001  A INIT   RMB     1         GEN PURPOSE FIRST PASS FLAG
01295A 8133      0001  A ERROR  RMB     1         USED IN PLOAD FOR CHECKSUM ERROR
01296A 8134      0001  A ROLFLG RMB     1         ROLENT FIRST PASS FLAG
01297A 8135      0001  A FLG24  RMB     1         ROLENT 2/4 NIBBLE MODE FLAG(CLR=4)
01298A 8136      0001  A FNCFLG RMB     1         FUNCTION FLAG(FUNC SET/CLEAR)
01299A 8137      0001  A INIT2  RMB     1         FRESH 2ND START FLAG
01300A 8138      0001  A LDFLAG RMB     1         USED IN PLOAD FOR I/O DIRECTION
01301            8139  A LDTBLE EQU     *         TABLE USED BY PLOAD FOR ADDR
01302A 8139      0001  A BCOUNT RMB     1         HOLDS BYTE COUNT FOR PLOAD
01303A 813A      0002  A BEGA   RMB     2         HOLDS BEGINNING ADDR FOR PLOAD
01304A 813C      0002  A ENDA   RMB     2         HOLDS ENDING ADDR FOR PLOAD
01305A 813E      0002  A POINTR RMB     2         USED IN BKPT ROUTINE
01306A 8140      0002  A TOFT   RMB     2         USED IN BKPT
01307A 8142      0001  A NUMBER RMB     1         .  "
01308            8143  A BOF    EQU     *
01309A 8143      0010  A        RMB     16        BREAKPOINT TABLE
01310            8153  A EOF    EQU     *
01311A 8153      0010  A        RMB     16        AUX BREAKPOINT TABLE
01312                           END
TOTAL ERRORS 00000
```

```
800A ACIA0  00106*00993 01069 01149 01173 01209
800B ACIA1  00107*01248 01250
FCF6 ACTION 00847 00856*
FD09 ACTON2 00857 00863*
FFD8 ADDR   01184 01221 01240*
F94D ADDXA  00324*00381 00639 00732
FA6A ADV1   00483 00489*
FA7D ADV2   00490 00497*
FA96 ADV3   00498 00507*
FAA6 ADV4   00508 00514*
FAB6 ADV5   00515 00521*
FC61 ALDUN  00770 00778*
FAD9 ARN1   00537 00539*
FCCA ARN17  00829 00832*
F9F6 ARN1M  00435 00438*
F95B ARND   00327 00329*
F8F7 ARND1  00272 00274*
FBEF BADOFS 00687 00690 00694 00698*
FCA5 BADSTK 00808 00818*
8139 BCOUNT 01167 01302*
813A BEGA   01003 01169 01195 01303*
FAF4 BEGIN  00553 00557*
F8E7 BKP    00267*
F901 BKPCNT 00279*
FA3C BLANK  00464 00467*
8143 BOF    00130 00851 00859 00871 00879 00908 00909 00916 00956 00958 00971
FB63 BOT2   00610 00621*
FB75 BOT4   00612 00629*
F9A2 CLOP   00373*00374
FA47 CLR1   00471 00473*
FA45 CLR14  00469 00472*
F846 CLRD   00167*00182 00818 01054 01070
F848 CLRDSP 00168*00473 00648 00673 00776 00914 01027
F97F COLFND 00349 00354*
F971 COLUMN 00347*00352
FCDA CONT   00843 00846*
F8CC CONTIN 00212 00252 00255*
8115 DIGEN  00295 00296 00309 01276*
811C DISBUF 00168 00173 00185 00298 00461 00466 00675 00727 00767 00820 00822
8089 DISCTR 00101*00211 00230
F8A4 DISNMI 00140 00213 00227*
FB1A DISP   00562 00569 00575*
FBB9 DISP1  00601 00673*
FE75 DISP10 01024 01026*
FE73 DISP11 01008 01025*01088
FE66 DISP2  01004 01020*
FD73 DISPLY 00855 00908*00955
FD7F DISPY  00867 00885 00907 00912*
FD7C DISPY1 00862 00911*00957
8088 DISREG 00100*00138 00210 00303 00305 00345
F949 DLY    00316 00318*00319
F946 DLY1   00304 00317*
F941 DLY25  00315*00384
FB93 DONE   00647*
FBB5 DSP    00671*
FB67 DUBROL 00614 00623*
FA91 DUNADV 00505*00525
FA16 DUNPAS 00448 00452*00506
```

```
    FC08 DYS     00577 00707*
    FC0B DYSCOD 00457 00646 00671 00707 00713*00912 01026
    FC3C DYSTBL 00730 00745*
    F839 EFPI    00146 00153*00306 00789
    FE1B END     00961 00973 00986*
    813C ENDA    01018 01197 01218 01220 01304*
    FD87 ENDBPT 00891 00894 00915*00935 00937
    FEA0 ENDFST 01046 01049*
    FC0A ENDOFF 00678 00708*
    F8AF ENNMI   00199 00221 00236*
    8153 EOF     01310*
    FEEB ERR     01083 01087*01128
    8133 ERROR   01055 01066 01087 01295*
    8130 EXFLAG 00147 00218 00293 00342 00582 00791 01291*
    F008 EXGET   00111*00112 00344 00584
    F00E EXKEY   00113*00292
    F00B EXPUT   00112*00113
    F005 EXREEN 00110*00111
    F000 EXROM   00108*00109 00160
    F002 EXSTRT 00109*00110 00148
    F83F EXTST   00144 00160*00290
    FC59 FCLR    00661 00775*
    FDAD FCLR33 00888 00934*
    8135 FLG24   00504 00524 00555 00599 00609 00613 00705 00865 01000 01297*
    FBC2 FNC     00669 00677*
    FB79 FNCDCD 00197 00635*
    8136 FNCFLG 00643 00769 00777 00796 00842 00995 01298*
    FD88 FNDBRK 00251 00893 00916*00936
    FB9E FNHASH 00637 00656*
    FDAC FOUND   00925 00932*
    FC4C FSET    00660 00766*
    FD41 FSET33 00870 00887*
    FE7F FSTCLR 01009 01030*01065
    FC76 FTST    00793 00796*
    FD16 FUNCTN 00864 00868*
    FB0F FUNDIS 00556 00565 00571*
    FB01 FUNKEY 00558 00563*
    F964 GET     00220 00342*00353 00364
    F96C GET1    00343 00345*
    FAC4 GETIT   00530*00586
    FC64 GO      00280 00659 00789*
    FE4B GO00    01006 01009*
    FEC0 GO1     01034 01066*
    FCB6 GOEXIT 00817 00824*
    FB0A GOKEY   00568*
    FBE6 GOOD    00692 00694*
    FBE5 GOODOF 00688 00693*
    FC8A GOTO    00803*00845
    810A GTEMP   00355 00358 01270*
    810B GXTMP   01271*
    FB80 HASCLC 00638*00651
    FB8A HEX     00636 00643*
    8118 HEXBUF 00247 00250 00454 00456 00477 00478 00487 00493 00495 00501 00503
                00573 00576 00611 00615 00621 00624 00625 00628 00629 00683 00685
                00794 00854 00861 00882 00884 00896 00898 00906 00911 00920 00921
    FDE1 INBKPT 00802 00958*
    FF48 INCHR   01074 01078 01081 01084 01095 01102 01112 01114 01121 01148*
    FF49 INCHR2 01149*01151
```

```
FDA1 INCX    00923 00926*
8132 INIT    00263 00428 00430 00552 00554 01294*
812D INIT1   00129 00846 00849 01288*
8137 INIT2   00191 00548 00595 00597 00706 00856 00858 00997 01001 01014 01017
FE41 INPUT   00998 01005*
FFEE INPUT1 01247 01250*
FFE4 IRQ     01153 01246*
FFF1 IRQEND 01249 01251*
FFF8 IRQV    01255*
8117 KEY     00383 00533 01278*
F9A6 KEYCOD 00375*
FEB1 KEYM    01060*01090
F899 KEYNMI 00215 00218*
F9BF KEYTBL 00380 00392*
FB22 KEYTST 00438 00557 00582*00635 00668 00863 01005 01063
FED3 L1      01074*01076
FED9 L2      01077*01086
FEA1 LD      00996 01054*
8138 LDFLAG 01067 01172 01246 01300*
8139 LDTBLE 01110 01131 01240 01301*
FED3 LOAD    01073*
FDE6 LOOP    00960*00970
FFBD LOOP00 01221*01235
F869 LOOP1   00189*00192
F8CF LOOP10 00256*00260
FD99 LOOP1M 00922*00930
FDB8 LOOP20 00939*00947
FE00 LOOP40 00972*00985
FB69 LOOPM   00624*00627
F848 LOP     00169*00174
FC0F LP01    00715*00726
FC25 LP02    00729*00738
F91B LP1     00296*00310
F924 LP2     00299*00301
F905 MAIN    00278 00281*
FADC MEMCH   00548*00656
FEF0 MESAGE 01056 01089*
8100 MNPTR   00198 00282 00307 00641 00799 01263*
FF25 NEXT1   01118 01121*01132
FF36 NEXT2   01125 01130*
812F NFLAG   00188 00264 00281 00445 00828 00844 01290*
FFFC NMIV    01257*
F8A6 NMIX    00228*00237
F87F NONMSK 00207*01257
FC87 NORMGO 00797 00802*
FDAA NOTFND 00919 00927 00931*
8142 NUMBER 00850 00860 00883 00889 00904 00905 00918 00952 00953 01307*
FA13 NUMKEY 00439 00451*
FE91 NXT     01036 01041*
FE88 NXT0    01030 01035*
FE9A NXT1    01042 01045*
FB06 OFF     00563 00566*
FB32 OFFSET 00549 00567 00595*
FBAE OFST    00596 00668*
FDC9 OUT     00940 00948*
FA8E OUT1    00504*
FABF OUT2    00488 00496 00524*
FFC3 P1      01222 01224*
```

```
FF5E P2      01164*01168
FBF4 PAST    00697 00700*
FA23 PAST1   00453 00457*
FA32 PAST2   00459 00463*
FE7B PEND    01028*
FF0B PIN     01080 01104 01110*
FE1C PLOAD   00662 00991*
FE2C PNCH    00997*
FF40 PNLDR   01139*01142 01183 01217
FF3D PNLDR1  01138*01194 01202
813E POINTR  00852 00868 00880 00900 00917 00962 00966 00974 00982 01305*
FF54 POUT    01159*01193 01231
F871 PROMP1  00196*00778
F861 PROMPG  00186*00823
F857 PROMPT  00145 00182*00219 00437 00538 00657
FFA7 PUNCH   01029 01208*
FE7C PUNCH1  01015 01029*
FF7E PUND10  01184*01198
FF84 PUND25  01185 01187*
F90B PUT     00200 00290*00311 00800
F913 PUT2    00291 00293*
FEEE QUIT    01064 01079 01088*01101
FFA1 QUIT1   01201*01230
FF04 RDBLCK  01097 01102*
FA62 REGAD1  00432 00486*
FA4B REGADV  00436 00441 00449 00477*
F9DD REGDIS  00276 00428*00658
F9EA REGLOP  00429 00433*
8116 REGNUM  00431 00452 00458 00479 00481 00484 01277*
F800 RESTRT  00123*01258
FADB RET     00535 00540*
FE9E RET44   01038 01044 01047*
FDFD RMBKPT  00267 00971*
FB44 ROLENT  00451 00559 00606*00645 00670 00866 01007
8134 ROLFLG  00505 00574 00598 00606 00608 00704 00792 00848 00892 00938 01021
FB56 ROLL    00607 00613*
F987 ROW     00358*00367
812E ROWCOL  00187 00372 00433 00530 00532 00585 01289*
F99A ROWFND  00361 00368*
FF4F [illegible]  01152*01177
FFFE RSTV    01258*
FFB1 S1200P  01216*
FEF5 S300LD  01072 01095*01099 01105
FF79 S300P   01182*01211
8131 S3FLAG  00999 01037 01043 01071 01100 01210 01292*
8124 SAVSTK  01281*
806B SCNCTR  00103*00214 00228
806A SCNREG  00102*00136 00154 00183 00297 00385
FD31 SKIP    00874 00878 00880*
FB2A SKIP1   00583 00585*
F850 SKIP11  00170 00172*
FD27 SKIP2   00872 00875*
FA8B SOUT1   00503*00513 00520
817F STKTOP  00099*00123 00196 00262
FF34 STORE   01123 01129*
F8B3 SWINT   00244*01256
FFFA SWIV    01256*
FCD0 TBRK    00663 00842*
```

```
8085 TCNTRL 00104*00208 00831
8086 TMRMSB 00105*00128
8140 TOFT   00131 00853 00873 00877 00895 00903 00926 00939 00948 00951 00960
FA0A TRAC   00443 00447*
FA0E TRAC1  00446 00449*
FCD5 TRACE  00450 00844*
F8FB TRCE   00266 00276*
8126 UA     00500 00503 00826 01284*
8125 UB     00510 00516 00825 01283*
8124 UCC    00255 00259 00517 00521 00832 01282*
8102 UHASH  00650 01264*
F95F UIRQ   00334*01255
8106 UIRQV  00334 01266*
8104 UNMIV  00216 01265*
8129 UPC    00268 00274 00486 00491 00795 00798 00801 00809 00811 01120 01286
FB99 USERFN 00644 00650*
812B USP    00126 00261 00485 00522 00803 01287*
8108 USWIV  00253 00273 01267*
8127 UX     00492 00499 00824 01285*
8132 VERIFY 00991 01031 01122 01293*
FE14 WHOOPS 00979 00982*
FF6C WRITE  01140 01160 01162 01165 01171*01189 01191 01200 01226 01229 01234
FF70 WRITE2 01173*01176
8113 XCALC  00324 00325 00326 00328 00329 01275*
810D XTMP1  01272*
810F XTMP2  00729 00734 01273*
8111 XTMP3  00551 00679 00681 00684 00686 00701 00994 01020 01274*
```

APPENDIX 2

PARTS LIST

APPENDIX 2. PARTS LIST

Item Number	Quantity	Description	Part Number	Reference Designation
1	1	PC Board	MEK6802D3	-
2	1	Integrated Circuit	74LS01N	U7
3	1	Integrated Circuit	74LS10N	U12
4	1	Integrated Circuit	74LS20N	U27
5	1	Integrated Circuit	74LS28N	U1
6	1	Integrated Circuit	74LS27N	U13
7	2	Integrated Circuit	74LS138N	U10, U11
8	2	Integrated Circuit	74LS241N	J2, U3
9	4	Integrated Circuit	74LS244N	U4, U5, U6, L17
10	1	Integrated Circuit	MC6802P	U26
11	2	Integrated Circuit	MC6810P	U30, J31
12	1	Integrated Circuit	MC6821P	U8
13	1	Integrated Circuit	MC6846P	U9
14	1	Integrated Circuit	MC14539BCP	U28
15	5	Integrated Circuit	MC75452P	U14, U15, U16, U33, U34
16	1	Crystal, 3.579545 MHz	NDK, CTS, Buck-Man	Y1
17	1	Socket, IC , 16-Pin		(For SK1)
18	3	Socket, IC , 24-Pin		U29, U30, U31
19	8	Socket, 14-Pin		U18-U25
20	3	Socket, 20 Count		P1
21	3	Socket, IC, 40-Pin		U8, U9, U26
22	1	Resistor Array, 8-10K Ohm	Bourns, Beckman	U32

APPENDIX 2. PARTS LIST (cont'd)

Item Number	Quantity	Description	Part Number	Designation
23	1	Resistor, SIP, 3.3K Ohm	Bourns,Beckman	Z2
24	3	Resistor, 3.3K Ohm 1/4 watt, 5%		R1, R6, R7
25	4	Resistor, 10K Ohm 1/4 watt, 5%		R2, R3, R4, R5
26	1	Capacitor, 100 Micro Farad Electrolytic		C7
27	2	Capacitor, 27 Pico Farad DM - Dipped Silver Mica		C19, C20
28	24	Capacitor, 0.1 Micro Farad Monolithic Ceramic		C1-C6, C8-C16, C18, C21-C28
29	1	Capacitor, 1.0 Micro Farad Monolithic		C17
30	8	LED, 7-segment	Monsanto Man 74	U18, U25
31	1	Hex, Keypad	KB Denver-Custom	
32	4	Jumper, Zero Ohm Resistor		E2, E4, E5, E9
33	1	Key, Polarizing		(For P1)
34	1	Wafer Assembly, 5	Molex,Methode, Berg	(To ship with board)
35	1	Lens, PCB Display		
36	1	Test Point, Black (GND)		TP2
37	1	Test Point, Yellow (VCC)		TP1
38	2	Test Point, White (Signal)		TP3, TP4
39	3	20 pin Female connector	Molex	P1

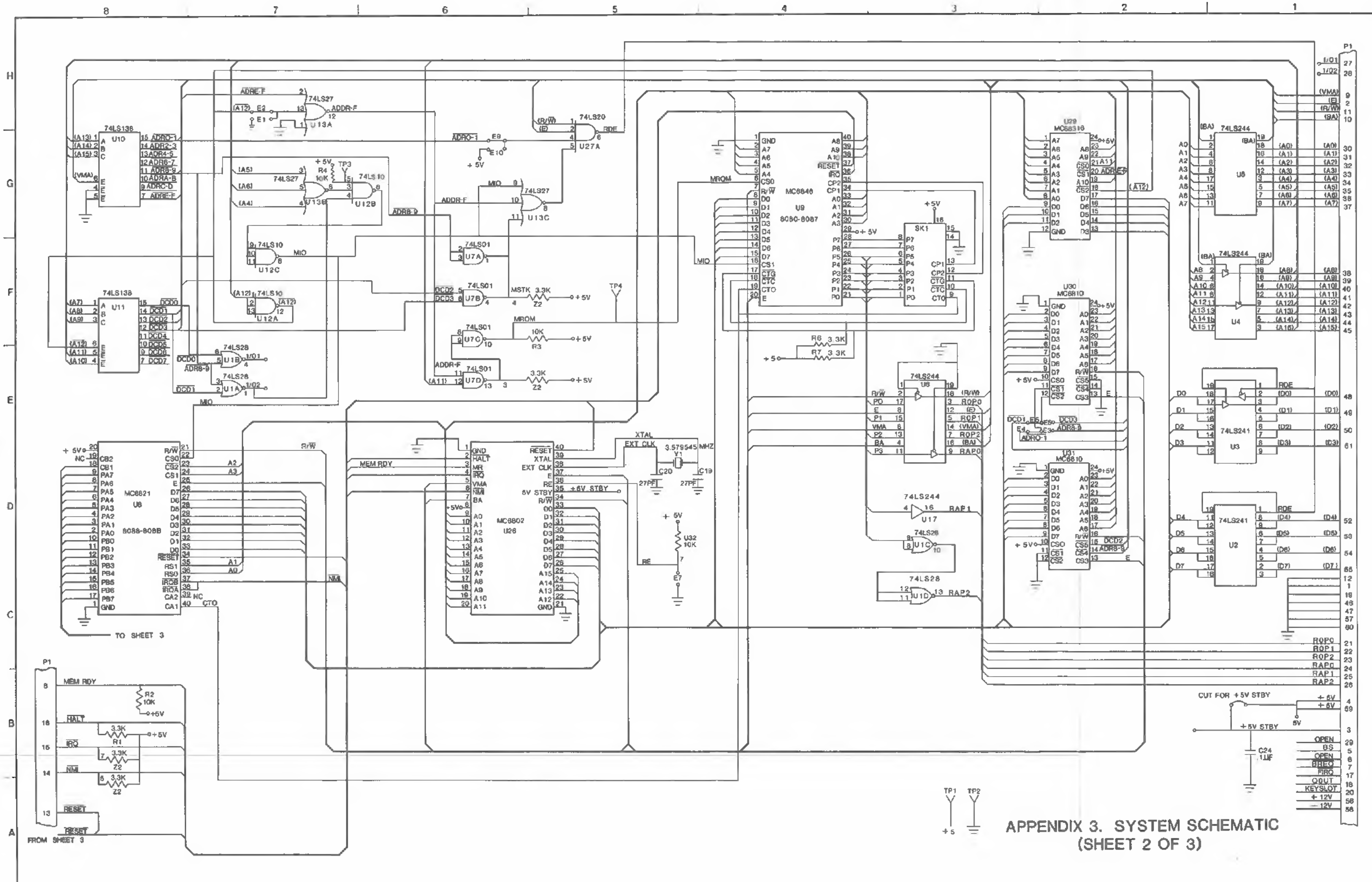
APPENDIX 3

SYSTEM SCHEMATIC

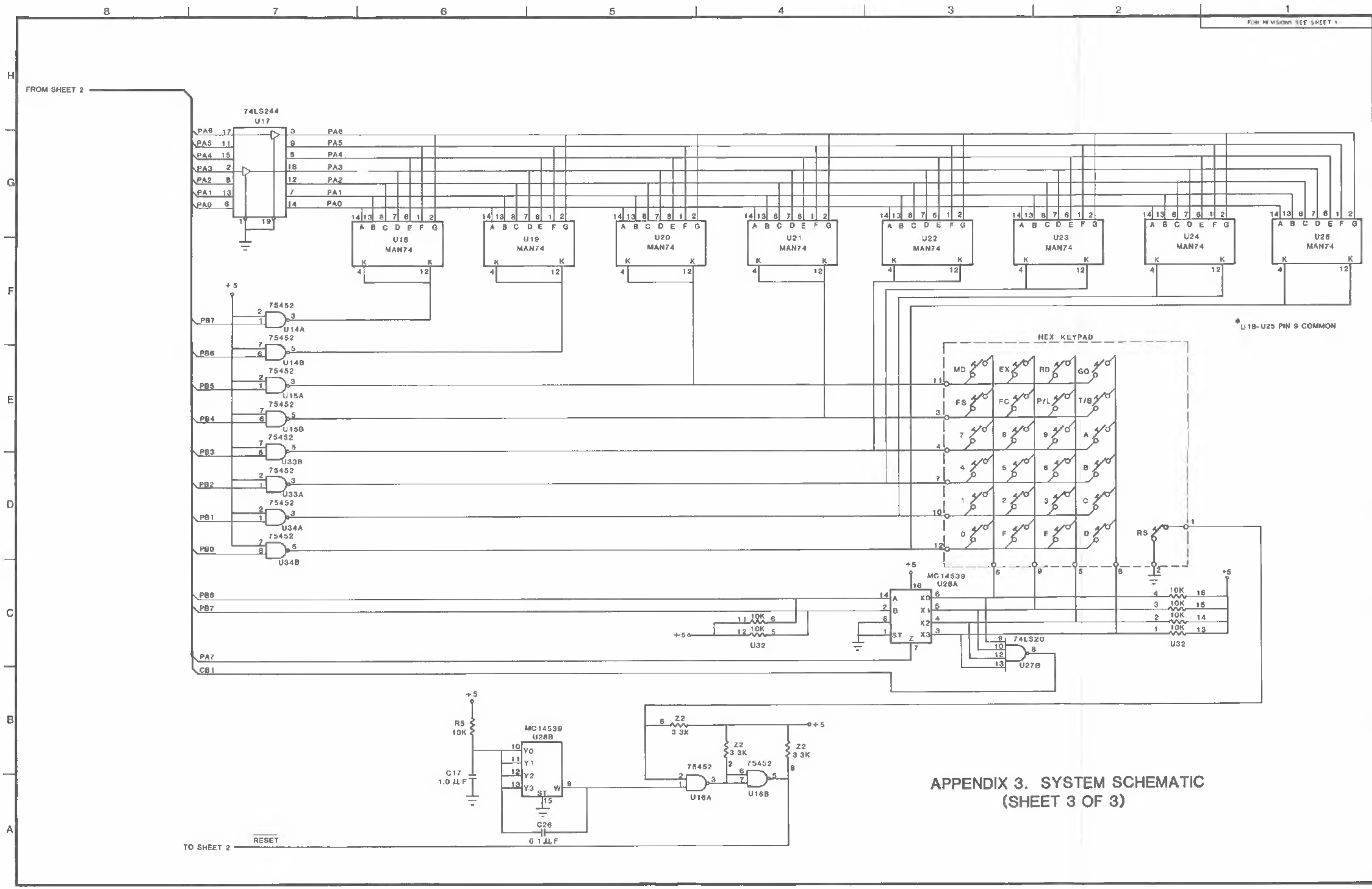
Sheet 1: Index Page

Sheet 2: System Schematic

Sheet 3: System Schematic



APPENDIX 3. SYSTEM SCHEMATIC
(SHEET 2 OF 3)



APPENDIX 3. SYSTEM SCHEMATIC
(SHEET 3 OF 3)

APPENDIX 4

DATA SHEETS

MC6802
MC6846
MC6821